软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(9):2402−2430 [doi: 10.13328/j.cnki.jos.005306]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

软件可靠性增长模型研究综述*

张 策[1,2], 孟凡超[2], 考永贵[3], 吕为工[2], 刘宏伟[1], 万 锟[2], 蒋家楠[1], 崔 刚[1], 刘子和[2]

[1](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

[2](哈尔滨工业大学(威海) 计算机科学与技术学院,山东 威海 264209)

[3](哈尔滨工业大学(威海) 理学院,山东 威海 264209)

通讯作者: 张策, E-mail: zhagnce@hitwh.edu.cn, zhangce2002@126.com

摘 要: 软件可靠性增长模型 SRGM(software reliability and growth model)是目前建模可靠性及其过程提高的重要数学工具,对可靠性的评测、保证以及测试资源管控和最优发布研究具有重要作用.对 SRGM 的核心研究内容与建模流程进行分析,给出了 SRGM 基本功用.同时,梳理了 SRGM 的发展演变历程,进而对当前研究现状进行深入剖析,给出当前研究特征.从软件中总的故障个数、故障检测率 FDR(fault detection rate)和测试工作量 TE(testing-effort)这 3 个方面对影响 SRGM 的因素进行了分析.基于作者前期研究中提出的统一性框架模型,对当前典型的解析模型进行了分类比较和分析;对基于有限与无限服务队列模型的 SRGM 进行分析与讨论;对以率驱动事件过程 RDEP(rate-driven event processes)为重点的仿真方法进行剖析.进一步地,为了验证与分析不同模型的差异,对 26 个典型的模型在公开发表的 16 个数据集上进行了实验.结果表明,SRGM 的性能差异取决于失效数据集的客观性以及研究人员对测试过程进行不同假设下所建立的数学模型的主观性.最后,指出了 SRGM 面临的挑战、发展趋势和亟待解决的问题.

关键词: 软件可靠性增长模型;不完美排错;测试工作量;框架模型;排队论;仿真
中图法分类号: TP311

中文引用格式: 张策,孟凡超,考永贵,吕为工,刘宏伟,万锟,蒋家楠,崔刚,刘子和.软件可靠性增长模型研究综述.软件学报, 2017,28(9):2402−2430. http://www.jos.org.cn/1000-9825/5306.htm

英文引用格式: Zhang C, Meng FC, Kao YG, Lü WG, Liu HW, Wan K, Jiang JN, Cui G, Liu ZH. Survey of software reliability growth model. Ruan Jian Xue Bao/Journal of Software, 2017,28(9):2402−2430 (in Chinese). http://www.jos.org.cn/1000-9825/5306.htm

Survey of Software Reliability Growth Model

ZHANG Ce[1,2], MENG Fan-Chao[2], KAO Yong-Gui[3], LÜ Wei-Gong[2], LIU Hong-Wei[1], WAN Kun[2], JIANG Jia-Nan[1], CUI Gang[1], LIU Zi-He[2]

[1](School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

[2](School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai 264209, China)

[3](School of Science, Harbin Institute of Technology at Weihai, Weihai 264209, China)

Abstract: SRGM (software reliability and growth model), as an important mathematical tool of modeling reliability and improving reliability process, plays significant role in measuring, predicting and ensuring reliability, managing testing resources and releasing optimal software. Research on SRGM is elaborated and analyzed in this paper. First, main research content and modeling process of

* 基金项目: 国家科技支撑计划(2014BAF07B02, 2013BA17F02); 国家自然科学基金(61473097); 山东省自然科学基金(ZR2015FM006); 山东省科技攻关项目(2011GGX10108, 2010GGX10104)

Foundation item: National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2014BAF07B02, 2013BA17F02); National Natural Science Foundation of China (61473097); Natural Science Foundation of Shandong Province of China (ZR2015FM006); Key Science and Technology Program of Shandong Province (2011GGX10108, 2010GGX10104)

收稿时间: 2014-10-14; 修改时间: 2016-02-09, 2016-08-15; 采用时间: 2017-04-24; jos 在线出版时间: 2017-06-05

CNKI 网络优先出版: 2017-06-05 16:32:46, http://kns.cnki.net/kcms/detail/11.2560.TP.20170605.1632.004.html

SRGM are analyzed, and the basic function is sketched. In the mean time, research evolution is summarized, the state of art is illustrated and the current research characteristics are formulated. Second, from the three aspects including the total number of faults in software, FDR (fault detection rate) and TE (testing-effort), the key factors influencing SRGM are analyzed. Based on the unified framework model proposed in author's previous research, the classical numerical models are classified, compared and analyzed. In addition, the SRGMs based on finite and infinite queue model are discussed and simulation technique emphasizing on RDEP (rate-driven event processes) is elaborated. Furthermore, to evaluate the differences in the models, 26 models are compared by 16 published failure data sets. Experimental results reveal that the differences depend on the objectivity of failure data set collected and the subjectivity of establishing mathematical model by researchers under the different assumptions. Finally, the challenges, the trend of development and the problems to be solved are pointed out.

Key words: software reliability growth model; imperfect debugging; testing effort; framework model; queue theory; simulation

准确建模软件可靠性并且预测其可能的增长趋势,对于确定整个产品的可靠性至关重要[1−5].软件可靠性增长模型 SRGM(software reliability growth model)提供了与软件代码紧密相关的可靠性特征信息[6],例如剩余故障数量、累积检测或排除的故障数量等.SRGM 在度量、预测、提高与保证可靠性上被广泛应用[7,8],同时,定量地对涵盖测试资源与成本管控和最优发布时间抉择[9−16]等在内的软件开发活动提供重要决策支持,是可靠性工程研究领域内的重要手段.

由于可靠性更多更直接地源于软件开发人员在设计与编制程序过程中所带来的内在错误(error),在测试阶段中,大量测试计划的有效实施可促使这些错误引发故障(fault),当达到一定条件时,故障会引发系统失效(failure).这样,SRGM 通过失效的检测能够促使开发人员回溯至代码的层面进行调试修改与排错,进而提高可靠性;同时, SRGM 在面对一段时间下的失效与改正故障数量可探究出当前测试环境下可靠性随时间的变动规律.正因如此,多年来,研究人员在 SRGM 上积累的技术框架在可靠性过程综合管理上可有效确保可靠性得到持续提高,尤其在测试阶段;同时,在测试资源分配、成本管控、发布策略上得到了应用上的广泛认可.

SRGM 对软件可靠性研究形成了重要的理论支撑,对有效解决可靠性动态定量增长问题开辟了途径.在具体研究可靠性增长的突破点与线索上,SRGM 以失效的观察、记录,故障的检测、移除为主线,采用微分建模的方法来描述(累积检测与/或修复的)故障个数与可靠性间的数学关联,并据此指导测试策略的优化执行,用以实现检测与排除掉更多的故障来提高可靠性.

经过 30 多年的发展,SRGM 已经取得了显著的进展,涌现出若干典型模型,已拓展出不同技术脉络,正处在承前启后的阶段,迫切需要对其研究工作进行系统化的梳理和评述.目前,国内外尚没有对 SRGM 进行全面述评的综述文章,本文在作者前期工作的基础上,搜集并综合分析了百余篇典型的 SRGM 文章来进行梳理,从建模思路、技术归类讨论、模型差异比较等视角对 SRGM 进行了全面述评,并基于此给出了后续研究的新挑战、趋势和亟待解决问题,以期为科研人员提供有价值的分析与参考,促进 SRGM 取得新的进展.

本文第 1 节对 SRGM 研究内容、基本功用进行阐释.第 2 节梳理 SRGM 的发展历程,对当前研究进展进行概要分析,给出研究所呈现的主要特征.第 3 节从影响 SRGM 的 3 个因素入手来分析.第 4 节基于我们所提出的统一模型进一步对 SRGM 的类别进行讨论,涵盖框架模型、队列模型和仿真方法.第 5 节对典型 SRGM 性能间差异进行比较分析.最后指出当前研究存在的挑战、趋势、亟待解决的问题,并给出结论.

1 核心研究内容、建模流程与功用

1.1 SRGM研究核心内容——以G-O模型为例

软件测试过程中,随着故障不断被检测出来并被排除,进而使得软件可靠性持续获得增长,这为可靠性研究提供了有效的切入点.SRGM 从软件失效的角度进行可靠性建模,采用以微分方程(组)为主的数学手段建立软件测试过程中的若干个随机参量间的定量函数模型,例如测试时间、累积检测的失效或修复故障个数、测试工作量 TE(testing-effort)等参量.基于求解获得的累积检测故障数量函数表达式(通常以 $m(t)$作为标记),可以获得测试阶段的可靠性.因此,建立准确的能够描述真实随机测试过程的累积检测故障数量函数 $m(t)$,成为了 SRGM

研究的关键.

面对自故障检测至排除的软件测试过程,基于不同的假设建立形式各异的数学模型产生不同的结果.从流程抽象的视角,不同 SRGMs 的获取总可以被定型地概述为如图 1 所示给出的研究过程.

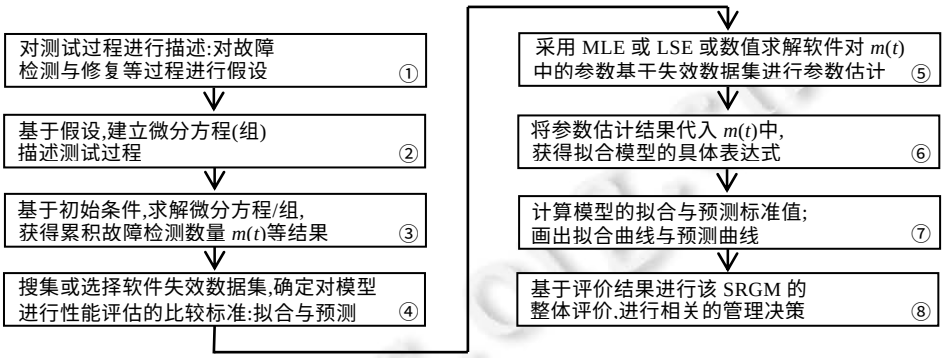


Fig.1 Research process of the SRGM

图 1 SRGM 研究过程抽象

可以看出:SRGM 研究的机理是对测试过程中各随机变量间建立适当的数学模型,用以指导测试资源的消耗,以便在测试周期内提高可靠性.数学模型的有效性,主要通过在真实的失效数据集上进行拟合与预测两个方面上进行验证.由于研究人员对测试过程的认知存在差异性,因此,SRGM 研究已有众多模型被提出.在当前研究中,非齐次泊松过程 NHPP(non-homogeneous Poisson process)类 SRGM 最具有吸引力,应用也最为广泛[17−20],现有上百个 NHPP 类 SRGM 的建立均包含下述基础公共假设[5,21−31]:(1) 假设失效事件随机发生,测试人员与排错人员对失效的观察与故障的排除满足 NHPP;(2) 设$[N(t),t\geq 0]$为随机计数过程,$N(t)$为$[0,t]$内测试人员累积检测的故障数量,且 $\mathrm{E}[N(t)]=m(t)$,其中,$m(t)$为均值函数,满足 $m(0)=0$.则利用 NHPP 基本性质,可以得到:

$$\Pr[N(t)=k]=\frac{[m(t)^k]\times \mathrm{e}^{-m(t)}}{k!},k=0,1,2,\ldots \tag{1}$$

$$m(t)=\int_0^t \lambda(\tau)\mathrm{d}\tau \tag{2}$$

测试阶段的软件可靠性 $R(x|t)$可表示为

$$R(x|t)=\mathrm{e}^{-(m(t+x)-m(t))} \tag{3}$$

若设 $t(t\geq 0,x>0)$为上次失效发生的时间点,则软件可靠性在$(t,t+x]$内可表示为公式(3).

这里,我们以经典的 G-O 模型为例,对 SRGM 的建立进行简要介绍,在第 4.1 节中,我们给出的研究分类将进行更为深入的剖析与阐释.G-O 模型的建立是基于假设:(1) 软件失效满足 NHPP;(2) t 时刻,累积检测到的故障数量 $m(t)$变化率与当前剩余的故障数量 $a-m(t)$成正比例,比例系统为 b.则可得到下面的微分方程:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b\times(a-m(t)) \tag{4}$$

其中,a 为软件中总故障个数.在 $m(0)=0$ 的初始条件下,可求得 $m(t)=a(1-\mathrm{e}^{-bt})$.

SRGM 的关键是在某些假设条件下建立微分方程,求解得到 $m(t)$,而 $m(t)$的相互间差异是不同 SRGM 的最直接体现.由公式(3)可以看出,$R(x|t)$是 $m(t)$的函数.因而可靠性随时间的增长完全取决于 $m(t)$.而 $m(t)$与真实失效个数的拟合与预测情况(即接近情况)是衡量 SRGM 性能优劣的根本标准.这样,获得特定测试环境下合适的 $m(t)$是 SRGM 研究的核心内容.

1.2 基本建模流程与功用

图 2 给出了科研人员基于对测试过程的认知,进行 SRGM 建模的基本流程以及 SRGM 的功用.由图 2 可以看出,其中的步骤①是测试过程的流程描述,基于对测试过程中获得的相关数据(失效时间或间隔、失效个数、

测试工作量 TE 等),研究人员通过数学建模与统计分析等手段,充分挖掘出测试过程中各因素间的数学定量关系(例如,累积检测的失效个数 $m(t)$与测试时间 t 的关系),用以度量和评测历史测试情况,以及预测后续测试情况.另外,测试阶段中对测试资源消耗的掌控,并由此引出的软件发布问题,都是基于 SRGM 研究作为支撑.

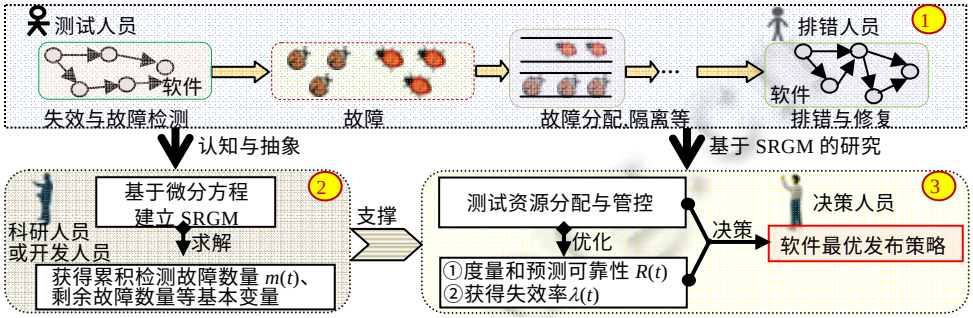


Fig.2 Modeling flow and function of the SRGM

图 2 SRGM 建模流程与基本功用

真实测试环境的不稳定直接导致了 SRGM 需要把实际的随机因素考虑进去[32−34],从关注失效个数的发现直至故障被排除的角度来提高软件可靠性,是 SRGM 用以建模描述软件测试过程的最主要线索.SRGM 旨在探索软件测试过程中可靠性同与其直接相关的决定因素间的内在机理——故障检测至排除过程对可靠性的定量影响关系,为可靠性提高、确定成本结构和软件最优发布等提供决策依据.

2 发展历程概述与当前研究现状分析

2.1 发展历程概述分析

软件工程师在从事软件制品的生产开发活动中带有大量复杂的随机性与不确定因素,在此背景下,SRGM 研究持续至今.表 1 与图 3 概要性地描述了 SRGM 的发展历程.

Table 1 Interpretation of Fig.3

表 **1** 对图 3 的概要解释

类型	各类型发展阶段概述
经典模型 C	C①: NHPP 类 SRGM 的起源为指数型的 G-O 模型[22],其后,Delayed S-shaped[35,36]和 Inflection S-shaped[37]以及 Yamada 模型[38]相继被提出,这些模型均是早期研究 SRGM 的典型模型.其假设条件虽偏离真实测试情况程度较大,但却成为后续研究的重要参考和对比对象,在 SRGM 研究体系中占有重要地位,因此这里称为经典模型; C②: 在此之后的研究中,逐渐放宽假条件,使得研究持续深入;文献[39]提出了高斯白色噪音函数与指数函数相结合的 FDR 函数,在 8 个数据集上得到了验证,但本质上依旧是完美排错模型.
不完美排错 ID	ID①: 这一时期主要从新故障引入或不完全排错进行研究[38,40−43],但其中,Schneidewind[44,45]将故障修复延迟视为检测的后续过程,建立了考虑修复延迟(即改正过程)的可靠性模型; ID②: 通常包含不完全排错与引入新故障以及其他更多的实际情况[24,46−48].
考虑 TE	TE①: 最早于 1986 年[49]SRGM 研究中出现了 TE(testing-effort),但直至 20 世纪 90 年代中期,二者的结合研究并不深入; TE②: 进入 21 世纪,各种类型的 TE 开始广泛地出现在 SRGM 中[5,6,25,29,30,48,50−56,57],使得融入测试资源的花销到建模中成为常态,例如,文献[57]将 S 型 TEF 融入到不完美排错下的可靠性建模中,取得较好效果.
考虑 CP	CP①: 此为 CP 研究的早期,主要是进行 CP 相关的 SRGM 中参数估计[58,59],寻找 CP 的位置[60],将 CP 与不完美排错或 TE 结合[25,61]; CP②: 将多 CP 与 Weibull 类型的 TEF 相结合研究[31],以及从故障检测率的角度进行研究[26]; CP③: 将 CP 研究应用到墨西哥市臭氧峰的估计中[62]; CP④: 将 CP 与不完美排错、故障减少因子和成本等进行混合研究[63,64].

Table 1 Interpretation of Fig.3 (Continued)

表 1 对图 3 的概要解释(续)

框架模型 F	F①: 最早于 1985 年提出从 Bayesian 视角可以对 SRGM 进行统一描述[65]; F②: 证明以往的 SRGM 能通过 Self-exciting point 过程进行统一[66]; F③: 应用 3 种加权平均(加权算术,加权几何和加权调和平均)推演出现存的几种 NHPP 类型 SRGM[67]; F④: 从时间延迟角度以框架技术研究了故障检测与改正过程[68]; F⑤: 以统一的无限服务队列模型处理 3 种类型故障[69]; F⑥: 基于卷积运算提出了统一的不完美排错框架模型[24]; F⑦: 提出了考虑不完全排错与引入新故障的框架模型[48]; F⑧: 建立了基于 SRGM 进行可靠性的评估框架[1]; F⑨: 在不完美排错下,提出了统一的囊括故障检测与修复的软件可靠性建模方法[70].
仿真技术 S	S①: 最早基于率的仿真[71]; S②: 作为突破传统解析技术的重要方向和拓展,仿真技术得到了快速发展,实现了对不完美排错环境[72,73]以及排错人员和成本对测试过程影响的综合仿真[74,75].
队列理论 Q	Q①: 开始出现用无限服务队列模型来研究 SRGM[76,77]; Q②: 自 2008 年起,在 Huang CY[18,78,79],Lin CT[75]和 Chu TL[72,73]等人的推动下,基于有限与无限队列模型的 SRGM 混合研究在 CP、不完美排错、人员与成本分析方面取得进步.
发布问题 R	R①: 最早起源于 1980 年[80]; R②: 从综合考虑成本、TE 与测试效率的角度,对最优发布时间进行了研究[27,81]; R③: 从大数定理角度,分析了软件成本和最优发布中的随机性与不确定性[82]:实际成本可能大于期望成本; R④: 对考虑 TE 时的不完美排错下的成本构成与最优发布进行研究[56],并全面回顾了测试资源与成本管控和最优发布问题[9]; R⑤: 在不完美排错与担保成本下进行最优发布研究[15]; R⑥: 针对不完美排错下延迟型离散 SRGM 进行了最优发布研究[83]; R⑦: 提出将故障检测与修复过程综合考虑的多版本(连续更新)开源软件的最优发布模型[84],并于在线故障追踪系统(Mozilla 和 Gnome)上进行验证.

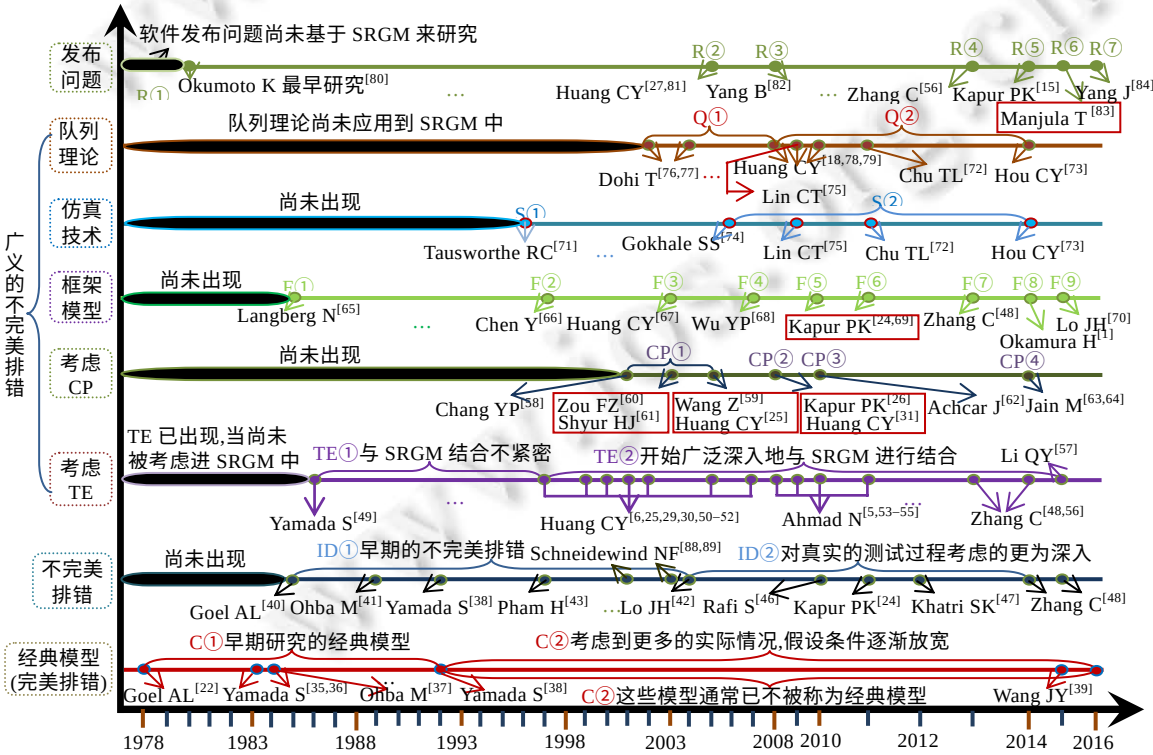


Fig.3 Overview of SRGM

图 3 SRGM 发展概览

2.2 当前研究现状简析

从检测与修复的故障数量角度来研究可靠性的增长,是 SRGM 研究的切入点.显然,被检测到的故障不断被排除,可靠性势必会得到提高.目前,与 SRGM 紧密相关的研究领域,整体上呈现了如下的技术演进路线:传统基于微分方程(组)建模测试过程中,故障检测至修复过程的解析方法(包含基于队列的研究)和基于仿真的分析方法(典型的,如离散事件仿真).在图 3 和表 1 的基础上,表 2 从多个研究角度选取了近年来典型的模型对 SRGM 的当前研究现状与趋势进行了概要剖析.

Table 2 Comparative analysis on research status

表 2 SRGM 研究现状比较分析

文献	假设条件与建模情况	求解方法	概述与分析	演化特征及使用范围
Ref.[21]	假设条件: ① 测试环境改变引发 CP;② 排错具有完全性,但会引入新故障定量方法:采用微分方程形式描述测试过程,并以 CP 的形式考虑到了测试环境的改变	复杂的解析方法	给出了当考虑 CP 时的多个典型 SRGM,进一步将研究引申至不完美排错环境下(仅考虑到了新故障引入)进行分析	突破了完美排错的限制,尚未建立较为全面的不完美排错模型: ① 用于考虑到测试策略与测试资源分配改变时的实际情况;② 考虑到新故障引入
Ref.[23]	假设条件: ① 考虑 TE 消耗;② 存在新故障引入现象 定量方法: 采用微分方程建立 考虑 TE 与新故障引入的模型	复杂的解析方法	建立了测试工作量相关的故障检测与修复过程,并考虑到新故障引入情况,进行了软件发布策略研究	从新故障引入角度建立考虑检测与修复 TE 的不完美排错模型: ① 用于考虑检测与修复测试资源消耗的场景;② 实施软件发布策略
Ref.[85]	假设条件: ① 将测试过程抽象为马尔科夫模型;② 存在新故障引入现象 定量方法: 随机过程理论	无法直接求解,只能给出上下界范围	从随机过程数学理论的角度进行推导不完美排错环境下的软件可靠性提高过程,给出了若干有价值的结论	将传统概率计算方法延展到随机过程理论: ① 聚焦在数学分析层面; ② 可用于理论推导给出基本范围
Ref.[72]	假设条件: ① 测试过程与排队系统类似;② 考虑到故障引入率 定性方法: 以排队论模型建模测试过程	离散事件仿真	建模故障检测与排错过程为单队列多通道队列系统,通过离散事件仿真模拟了不完美排错下的软件测试过程	将解析方法扩展到基于率函数的仿真方法: ① 用于模拟测试过程,预判故障数量、测试结束时间等; ② 辅助决策支持
Ref.[86]	假设条件: ① 排错的不完全性及新故障引入; ② 考虑到 TE 的消耗 定量方法: 利用 Cobb-Douglas 生产函数建立涵盖测试时间和测试覆盖效果的二维 SRGM	简单的解析方法	应用 SRGM 进行可靠性预测	将完美排错引申到从 Cobb-Douglas 视角下考虑到排错的不完全性与新故障引入的不完美排错:用于输入/输出模式下测试时间与测试覆盖率双重效用的场景
Ref.[87]	假设条件: ① 考虑 TE 的消耗;② 故障完全被排除,且不引入新故障 定量方法: 以微分方程的形式建立框架模型	解析方法	建立了考虑测试工作量与测试覆盖率的 NHPP 类软件可靠性建模框架	在完美排错的基础上建立考虑 TE 的框架模型:设定描述测试资源消耗的 TE 函数,进而获得特定模型
Ref.[24]	假设条件: ① 故障以概率 p 被排除,以概率 α 被引入;② 不考虑 TE 定量方法: 基于卷积和 Steiltjes 卷积,以微分方程建立含有不完全排错与新故障引入的测试过程	复杂的解析方法	建立了涵盖不完全排错与新故障引入的 SRGM 框架	将不完美排错引申到框架模型层面:用于框架内设定合适的参数、函数建立具体模型
Ref.[56]	假设条件: ① 排错具有不完全性且会引入新故障;② 考虑 TE 的消耗 定量方法: 以微分方程组的形式描述故障检测与修复	非解析方法:数值求解	基于不完美排错环境下 TE 相关的 SRGM 框架模型,建立了软件最优发布算法	完美排错和忽略过多实际特征不完美排错的扩展: ① 用于建立不完美排错时的可靠性模型;② 实施软件最优发布

整体上,基于解析方法的研究[5,21,23–31]依然占有非常大的比重,研究人员从对故障检测、排除的认识上出发,并考虑其他因素来建立数学模型.国内研究也取得了一定进展,文献[88–90]从测试与运行剖面的差异来研究SRGM,并在文献[91]中考虑到了故障的相关性问题.在不完美排错方面,文献[92]建立了不完全排错的 SRGM,并设定故障总数为随时间增长的函数形式,获得了良好效果;文献[93]在已知的两种不完美排错模型上,融入改进的 Logistic TEF 建立 SRGM,进一步,文献[94]又提出考虑 Logistic 型的测试覆盖率函数的 SRGM,并将研究工作拓展到文献[87]的框架模型上,推动了考虑 TE 的 SRGM 研究.仿真方法的出现和发展[72,74],使得研究人员可以回避求解复杂的解析模型而带来的困难.文献[73]针对不完美排错情况,给出了离散事件仿真下的故障检测与修复过程模拟,并在真实的数据集上进行验证,取得了较好的拟合效果.

综上,国内外对 SRGM 的研究已进入到多种技术并存的局面,经过我们的深入研究和梳理后,认为当前研究主要体现出下面几个特征.

(1) SRGM 的建立,促使定量研究可靠性的提高已成为目前可信性研究中体系化的一个分支,使得对失效的观察与预测、故障的检测与修复、可靠性的度量与预测、测试资源的分配与管控、发布问题的制定与决策形成了全面的研究分析支撑;

(2) 建立故障检测至排除过程的微分方程是 SRGM 研究中常采用的解析方法,这在 2007 年以前占有绝对的主导地位,至今也依然在延续.解析方法借助一个或多个微分形式的方程来公式化地建模描述累积检测与排除的故障个数以及考虑其他有关因素的数学关系式.通过求解可以获得累积检测的故障个数 m(t)和可靠性 R(t)等关系式,并由此展开深入的测试过程分析.但由于解析方法在求解复杂方程时会遇到较大困难:若研究的问题本身较为复杂,则相应的解析方法难以求解,此时,其适用的测试过程中的阶段数也是有限的;

(3) 应用排队论理论进行 SRGM 建模与研究是研究人员对测试过程深入认识的自然结果,是从传统基于微分方程研究框架下衍生的重要进步.队列模型更加符合真实测试过程中故障所经历的历程,因而得到了广泛应用[18,72,75,78],但更为深入地考虑排队论中更多的细节尚未触及.另外,队列模型目前主要用以建模故障修复过程,实际上也可以建模故障检测过程;

(4) 面对基于随机过程以及排队论研究引发的求解复杂问题,仿真技术[72,73]尤其是率控制下离散事件仿真是从解析方法研究范畴下进行突破的重要标志,这使得 SRGM 的研究体系呈现出非单纯数学因素的发展趋势更为明显.从 2007 年以来,仿真技术成为研究的热点,这其中,应用仿真方法对测试阶段软件可靠性进行相关研究已成为主流,这可以被解释为:软件测试过程本身是一个非常复杂的随机过程,是与多种因素相关的,这与仿真所具有的非线性特点类似,因而被广泛采用.仿真方法能够模拟对象或过程的主要关键特征,并为真实对象或过程的运行以及测试资源优化配置提供定量参考.仿真的最大优势在于避免复杂求解,且给出了更多的定性上的指导;

(5) 本文中,关于基于 SRGM 的扩展研究有两点特别指出:① 在我们的前期研究工作[9,56]中,已对测试资源与成本管控和软件最优发布问题进行了详细探讨,这里不再赘述;② 构件软件可靠性增长模型的研究最为明显的特征之一是把体系结构因素融入到 SRGM 中,目前主要还是基于单一软件形式的 SRGM 来累加拓展(将构件失效率$\lambda_i(t)$的某种累加结果作为整个构件软件的失效率$\lambda(t)$),且进展缓慢,在我们的前期研究成果[10]中对此有专门的分析,这里亦不再赘述.

3 影响 SRGM 的关键参数因素分析

文献[21]中明确指出,软件中故障总数 $a(t)$和故障检测率 $b(t)$是影响 SRGM 的重要参数因素.同时,经过我们的前期研究[9,48,56],认为测试工作量 TE(用测试工作量函数 TEF(testing-effort function)来表征,即 $W(t)$)也同样可归属于此.本节中,我们对影响 SRGM 的 3 个关键要素进行概要分析,为后续第 5 节中模型性能比较分析做必要的准备.

3.1 建模描述排错目标——总故障个数:$a(t)$

软件测试与排错人员希望将全部故障排除掉,获得高可靠性,但这并非易事也不现实.

(1) 首先,软件中总的故障个数是未知的(虽然可以假定为有限或无限个数),根本无法确定;

(2) 软件在排错过程中因排错的不彻底性以及引入新故障等现象,使得总故障个数会增加.

因而,设定 $a(t)=a$[36,67],忽略了排错中人为引入错误的事实.为此,设定 $a(t)$为 t 的某种增函数形式较为常见,我们将其归为表 3 中的 3 大类型.

Table 3 Analysis on the number of total faults in software

表 **3** 软件中总故障个数 $a(t)$分析

类型	软件中总故障个数 $a(t)$	概要分析
乐观类型:有限故障	$a(t)=c+a(1-e^{-\alpha t})$[43,95]	总故障个数有限:当 $t\to\infty$时, $\lim_{t\to\infty} a(t)=c+a$
悲观类型:无限故障	$a(t)=ae^{\alpha t}$[38,95], $a(t)=a(1+\alpha t)$[38,96], $a(t)=a(1+\beta t)$[97]	$a(t)$随测试时间 t 持续增长至无穷大
中间类型	$a(t)=a+\beta m(t)$[5,6], $a(t)=ae^{\alpha W^*(t)}$ [98]	由于 $m(t)$与 $W^*(t)$通常为增函数,但存在有限增长的可能,故 $a(t)$增长也可能存有限度

这里,有两点需要特别指出:(1) 悲观类型虽不易被接受,且根本无法在实际中验证,但有研究[77]指出,其在拟合真实失效数据上较乐观类型要好,这或许可解释为:对于实际软件,我们无法进行无限次的测试,因此软件中的故障个数无法准确估计;(2) 这些 $a(t)$考虑到了故障个数增加的情况,但直接导致其增加是由于排除故障过程所引发,因而,$a(t)$的变化率应与累积修复的故障数量变化率成正比,关于这一点,在我们的前期研究[48]中有阐释,在此不再赘述.

3.2 测试环境描述能力——故障检测率*FDR*(fault detection rate)

FDR 用以描述当前测试环境下故障被检测出的程度大小,其与整体测试策略(技术、工具、测试人员技能等)紧密相关.由于测试策略的多样性,使得 FDR 存在多种函数情况,且均为研究人员自行设定.

早期研究中认为 FDR 为常量,设定 $b(t)=b$.随着研究的深入,FDR 已呈现出多种函数形式,例如 $b(t)=b^2t/(1+bt)$[36,67], $b(t)=\dfrac{b}{1+\beta e^{-bt}}$ [96,99],$b(t)=bt^d+\delta\gamma(t)$[39],$b(t)=b\alpha\beta e^{-\beta t}(t)$[49], $b(t)=b\alpha\beta te^{-\beta t^2/2}$ [49]等.鉴于 FDR 是测试环境的直接描述,因而测试环境的改变,可借助 FDR 进行研究呈现.因而,当前对考虑变动点 CP(change-point)的 SRGM 研究中,多从建立 FDR 分段函数的形式来实施.

3.3 成本支持下的SRGM研究——考虑测试工作量TE

为实现预期发布,测试过程需要消耗测试资源作为保障.测试工作量函数 TEF 是对测试资源消耗的数学描述,表 4 列出了当前研究中主要的 TEF.

Table 4 Existing typical TEF

表 **4** 现有较为典型的 TEF

模型	公式化描述	解释
Logistic TEF[50]	$W_k(t)=\dfrac{W}{1+Ae^{(-\alpha kt)}}$	最早被提出的 Logistic 类型的 TEF
Generalized logistic TEF[27−29,81,100]	$W_k(t)=\dfrac{W}{\sqrt[k]{1+Ae^{(-\alpha kt)}}}$	依据参数 k 设置的不同,$W_k(t)$存在多种形式;若 k=1,则演变为 LogisticTEF 形式
	$W_{k,\beta}(t)=\left(\sqrt[k]{\dfrac{k+1}{\beta}}\Big/\sqrt[k]{1+Ae^{(-\alpha kt)}}\right)W$	若$\beta=k+1$,则 $W_{k,\beta}(t)$演变为上面的 $W_k(t)$
Deformable logistic TEF[56]	$W(t)=\left(\dfrac{e^{\alpha t}-v}{e^{\alpha t}+u}\right)W$	初值不为 0 的改进 Logistic 型 TEF
Weibull TEF[5,53]	$W(t)=(1-e^{(-\beta t^{\delta})})^{\theta}W, W>0,\beta>0,\delta>0,\theta>0$	根据其中参数的不同设置,Weibull TEF 又可具体分为 5 种特定的类型
Log-Logistic TEF[101]	$W_{ll}(t)=\left[\dfrac{(\theta t)^{\delta}}{1+(\theta t)^{\delta}}\right]W$	初值不为 0 的 Log 型 Logistic TEF

为评价表 4 中 TEF 的性能,我们开发了 TEF 性能评价系统:TPES,其具备的功能描述如下.

Fig.4 Presentation of TPES

图 4 TPES 运行展示

(1) 支持用户输入和选择不同的 TEF、TE 数据集和评价标准;

(2) 可实现对 TEF 在数据集上的拟合标准的计算、拟合与预测曲线的绘制;

(3) 提供用户设定评价标准的权重等;

(4) 实现多属性决策下的 TEF 综合性能的计算与排序.

图 4 是 TPES 运行的界面展示,其可以对当前 TEF 进行有效的比较与评价.

基于 TPES,我们在公开发表的 4 个失效数据集:TE-DS_1[102],TE-DS_2[103],TE-DS_3[104]和 TE-DS_4[105]上进行了验证实验.结果表明:整体上,改进的 Logistic 型 TEF 和 Weibull TEF 性能最优.这可归因于 Logistic 通常具有 S 型变化趋势,以及 Weibull 的多参数所带来的适应性.

4 关键技术分类讨论与评述

4.1 解析方法——框架式建模技术下分类建模讨论

由于不同的 SRGM 差异较大,难以辨识,统一建模方法的提出旨在屏蔽这些差异,采用框架技术是近年来可靠性研究的一个主题.按照对实际测试过程中的认识,当考虑到多种可能的测试环境时,这些 SRGM 在建模中主要涉及不完美排错 ID(imperfect debugging),TE,CP 等.因此,一种倾向是建立相对统一的 SRGM 框架模型,使其能够涵盖多种现有的模型.事实上,不同 SRGM 的本质区别在于对测试阶段认识的差异上,具体表现为所建立的微分方程(组)的不同.

下面是我们前期工作中基于统一建模法所提出的框架模型[9,48,56]:

$$\begin{cases} \dfrac{\mathrm{d}m(t)}{\mathrm{d}t} = \dfrac{\mathrm{d}W(t)}{\mathrm{d}t} \times [b(t) \times (a(t) - c(t))] \\ \dfrac{\mathrm{d}a(t)}{\mathrm{d}t} = r(t) \times \dfrac{\mathrm{d}c(t)}{\mathrm{d}t} \\ \dfrac{\mathrm{d}c(t)}{\mathrm{d}t} = p(t) \times \dfrac{\mathrm{d}m(t)}{\mathrm{d}t} \end{cases} \tag{5}$$

该模型的建立是基于下面的假设条件.

(1) 累积检测的故障数量 $m(t)$变化率与当前 TE 消耗率及故障检测率 $b(t)$下剩余的故障数量成正比;

(2) 故障引入率与修复过程中修复率成比例,比例系数为 $r(t)$;

(3) 修复率与检测过程中的检测率成比例,比例系数为 $p(t)$.

显然,研究人员对测试过程的认识差异直接决定了所建立的微分方程(组)的不同,这样所求得的 $m(t)$等变量存在差异.公式(5)可以涵盖现有的多个 SRGM,据此作为评述线索,给出 SRGM 的一种分类.

4.1.1 SRGM 起源:经典模型

在公式(5)中当不考虑第 2 子式与第 3 子式、忽略 TE,认为检测率 $\dfrac{\mathrm{d}m(t)}{\mathrm{d}t}$ 与尚未检测到的故障数量成正比,且设定 $b(t)$与 $a(t)$均为常量时,可以得到经典的 G-O 模型:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b \times (a - m(t)) \tag{6}$$

易见,G-O 模型忽略了测试中较多的实际因素,但却是最早的 SRGM,同时期的其他某型在建模上也较为粗糙.

4.1.2 考虑实际因素的 SRGM:不完美排错 ID 相关的模型

若公式(5)中不考虑第 2 子式与第 3 子式、忽略 TE,且认为检测率 $\frac{\mathrm{d}m(t)}{\mathrm{d}t}$ 与尚未被检测到的故障数量 $(a(t)-m(t))$成比例,可以得到典型的不完美排错模型——Pham 模型[45],其建立的微分方程为

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b(t)\times[a(t)-m(t)] \tag{7}$$

同时,Pham 认为,$a(t)=a(1+rt)$,即,$a(t)$是测试时间 t 的线性增函数.这表明测试中会引入新故障,因而被称为不完美排错模型; $b(t)=\frac{b(1+\sigma)}{1+\sigma\mathrm{e}^{-b(1+\sigma)t}}$,其中,σ是反映测试人员的学习因子.

另一方面,当 $a(t)=a,b(t)=b$ 时,公式(7)演变为最早的 G-O 模型.由于 $a(t)$与 $b(t)$可灵活设置,这样,公式(7)是框架模型.例如,若令 $b(t)=b\left(\gamma+(1-\gamma)\frac{m(t)}{a}\right)$ [106],其中,γ被称为弯曲因子,表示软件中不相关故障所占的比例.当 $\gamma=1$ 时,所得到的 $m(t)$即为指数型(exponential)SRGM,其余情况得到的是 S 型模型.这种转化现象被称为柔韧性,通常,框架模型都具有这种柔韧性.

4.1.3 融合 TE 的 SRGM:TEF 相关的 SRGM

故障检测与修复是在 TE 的消耗下进行的,TE 可有效地描述软件开发过程中的工作剖面[81],因而 SRGM 中应考虑到 TE 的存在.

这样,在公式(7)的基础上,当考虑 TE 时,可以得到下面被众多文献所采用的模型[5−7,25,27−31,51,81,100,106]:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}\times\frac{1}{w(t)}=b(t)\times[a-m(t)] \tag{8}$$

通常,鉴于公式(8)中考虑到了 TE,因而使其能够将测试资源的影响纳入到 SRGM 中.

易知,由于 $W(t)=\int_0^t w(t)\mathrm{d}t$ 存在多种形式,公式(8)也是一种框架模型.

4.1.4 考虑不完美排错与 TE 的 SRGM:ID 与 TEF 相关的 SRGM

由于建模的简化需要,很多 SRGM 做了一些偏离实际的假设,若研究中的假设条件更加靠近真实的测试过程,则所建立的 SRGM 可被称为不完美排错模型.若公式(5)中不考虑第 3 个子式,且认为新故障引入率与检测率成正比,则得到 Huang 模型[6]:

$$\begin{cases}\frac{\mathrm{d}m(t)}{\mathrm{d}t}\times\frac{1}{w(t)}=b(t)\times[a(t)-m(t)]\\ \frac{\mathrm{d}a(t)}{\mathrm{d}t}=r\times\frac{\mathrm{d}m(t)}{\mathrm{d}t}\end{cases} \tag{9}$$

这里,认为故障修复概率为 100%,新故障会在检测过程中被引入.更进一步,在公式(9)基础上,设定故障检测率函数: $b(t)=b\left[h+(1-h)\frac{m(t)}{a(t)}\right]$,可得到 Ahmad 模型[5].

显然,所建立的不完美排错模型越靠近真实的测试环境,则模型就越准确;但同时,求解方法已开始过渡到复杂的非解析方法.

4.1.5 考虑变动点 CP(Change-Point)、不完美排错 ID 与 TE 的 SRGM:CP,ID 与 TEF 相关的 SRGM

与前述相比,这是一种较为全面的建模,不仅涉及不完美排错 ID,TE,还将测试环境的改变而引发的 CP 融入到 SRGM 中.受到多种因素的影响(运行环境、测试策略、失效密度以及资源分配等)[26],软件失效分布并不均匀,这就为 CP 的出现提供了客观因素.这样,在公式(5)中,当不考虑第 2 子式与第 3 子式,设定 $a(t)=a$,且从 $b(t)$的角度来考虑 CP,则得到 GLTECPM 模型[25]:

$$\begin{cases} \dfrac{\mathrm{d}m(t)}{\mathrm{d}t}\times\dfrac{1}{w(t)}=b(t)\times[a-m(t)] \\ b(t)=\begin{cases} b_1,\ 0\le t\le\tau \\ b_2,\ t>\tau \end{cases} \end{cases} \tag{10}$$

其中,τ是 CP.相比之下,文献[31]从 $W(t)$的角度来考虑多个 CP,得到下面的模型:

$$\begin{cases} \dfrac{\mathrm{d}m(t)}{\mathrm{d}t}\times\dfrac{1}{w(t)}=b(t)\times[a-m(t)] \\ W(t)=\begin{cases} W(1-\mathrm{e}^{-\beta_1 t}), & 0\le t\le\tau_1 \\ W(1-\mathrm{e}^{-\beta_1\tau_1}+\mathrm{e}^{-\beta_2\tau_1}-\mathrm{e}^{-\beta_2 t}), & \tau_1<t\le\tau_2 \\ \cdots \end{cases} \end{cases} \tag{11}$$

其中,被分段讨论的 $W(t)$为 Yamada 指数型 TEF:$W(t)=W[1-\mathrm{e}^{-\beta t}]$,当然也可以是其他类型的 TEF.

相比于测试用例作为故障移除周期单位的离散时间模型[44],上面的以时间 t 作为自变量的 SRGM 被统称为连续时间模型,且后者居多.

4.1.6 小结与讨论

这些模型的提出,极大地丰富了 SRGM 的研究,使得在缺少安全性和可信性增长模型的情况下,SRGM 成为唯一能够描述可靠性随时间提高的技术手段.这里有 3 点需要指出.

(1) 当前,对不完美排错的认识主要停留在排错的不彻底上和/或新故障的引入上.实际上,我们认为,上述两种情况只是真实测试环境中很狭义的不完美,在此之外是更多的广义不完美.因而,不完美排错的概念与研究范畴理应进行更大范围的扩展;

(2) 虽然这些模型最终求解得到的 $m(t)$形式各异,但均是基于对测试过程进行不同假设后建立方程求解的结果.从该角度来看,这种解析方法就会存在着一个固有的不足:数学方程只能在有限范围内进行建模,因为复杂模型既难以提出又难以求解,因而新的技术手段,例如仿真,就在此背景下应运而生;

(3) 上述模型对测试过程的认识虽各有不同,但均只是局限在单纯的采用微分方程(组)来建模并求解范畴内.下一节采用排队思想建模测试过程,则使得研究人员对 SRGM 的认识更为深入.

4.2 排队论技术——建模实际测试过程的深入研究

研究人员认为[18,72,75,78],故障被陆续检测出来,按照某种顺序经历修复,该过程类似于排队等待的顾客与服务窗口(人员)间的关系,因而基于排队论的思想来研究软件测试过程,成为当前流行的趋势选择.

4.2.1 基于队列模型的排错行为建模与分析

使用队列模型来建模软件排错行为,通常基于如图 5 所示中的框架图.在该单队列多通道系统中[72,75,76,79],被检测出的故障以一定速率$\lambda(t)$到达,可按照一定原则(例如 FIFO 或其他优先级)进入等待队列,K 个服务窗口最多同时修复 K 个故障.下面的反馈虚线代表排错的不完美型,包括 3 种情况:① 被修复的故障没有被彻底排除,即,排而没除;② 排错中引入了新的故障;③ 既没排除又引入新故障.当前,软件的规模与复杂度均呈现上升趋势,因而多队列多服务通道形式的排队系统能够建模和处理更为真实的情况.

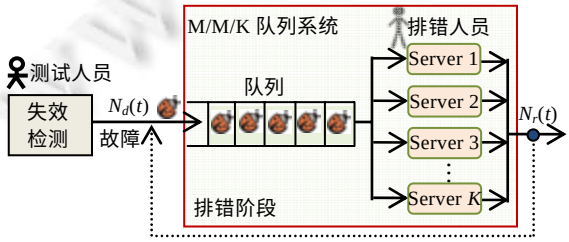


Fig.5 Software debugging queue: A single-queue multichannel debugging queuing system

图 5 典型软件排错队列:单队列多通道排错系统

针对无限服务队列模型 ISQ(infinite server queue)的不现实弊端,Huang[73]建模有限服务队列 FSQ(finite server queue)——不完美排错模型,通过在真实的数据集上进行可靠性的度量与预测,获得了较为准确结果.其不足主要有 3 个方面:① 只考虑到故障检测与改正过程;② 忽略了排错资源的局限性;③ 不完美排错仅指代引入新故障.后来,Huang[18]又针对测试过程中存在 CP 现象采用队列思想进行了分析,给出了在考虑多 CPs 时的累积故障修复数量的推导过程.其不足是:① 为了求解上的简易,队列模型是 ISQ 而非 FSQ;② 排错是完美排错而非不完美排错.这种忽视不完美排错或对不完美排错认识考虑不深入的不足,在文献[72]中得到了深入研究,Chu 采用单队列多通道队列模型基于率的仿真对软件的排错过程进行分析.此外,文献[75]还借助排队论和基于率的仿真程序对排错人员的配置与成本进行了探讨.

4.2.2 延迟问题——由排队机制引发

早期经典的 G-O 模型以及 Yamada Delayed *S* 型(DSS)和 Inflection *S* 型(ISS)模型中均假定某时刻故障改正数量 $m_r(t)$与检测的 $m_d(t)$相等,因而并不存在延迟.显然,这些模型严重偏离了实际.有两种情况的延迟.

- 第 1 种即为故障修复的时间:故障在修复队列中的修复时间 X 服从指数分布(一种延时[18,78],分布函数:$F(x)=\Pr(X\le x)=1-e^{-\mu x}$,密度函数:$f(x)=dF(x)/dx=\mu e^{-\mu x}$,$\mu$为服务率).当然,这种延迟在无队列技术的研究中[23,68,107]也存在;
- 第 2 种为当被检测到的故障等待一段时间积累到一定数量后才被修复和真正修复的时间.

第 1 种情况较为简单,由排队引发的第 2 种延迟问题值得关注.故障的复杂程度影响着移除过程,Kapur[69]认为:失效发生至排除之间延迟的不同,表示了不同严重程度的故障.并将其分为简单、严重和复杂 3 种类型,如图 6 所示.

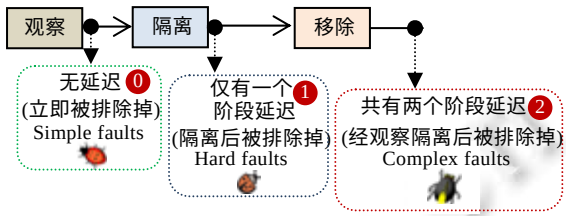


Fig.6 Simple, hard and complex faults

图 6 简单、严重和复杂这 3 种故障类型

被检测到的故障总是与某种严重级别相关[108],修复人员通常会以更高的优先权处理更加严重的缺陷故障,因而,文献[69]明确提出了三级故障类型,但在创建抢占式的队列模型上存有不足.此外,这些严重程度存有差异的故障间的关联也不能被忽视.到目前为止,在基于队列的研究中,考虑延迟问题的文献并不多见,但真实测试过程的这一环节显然是不能忽视的.

4.2.3 讨 论

综上,队列模型在 SRGM 研究中已有效展开,后续研究中需要考虑下面 3 个主要问题.

(1) 测试过程假设过于苛刻:失效发生后,检测人员检测出引发失效的故障,故障还需要被隔离与归类诊断分析.故障被指派到某队列后,因排错资源的局限性可能需要等待,排错人员也需要时间去排除故障,且可能存在不完美排错现象.上述过程是真实测试与排错环境下自然发生的,但在当前研究中却忽略了大量的相关细节,这使得虽然引入队列机制来处理排错,但严格的假设使其并没有明显体现实际测试特征;

(2) 缺乏多优先权队列与 SRGM 的集成研究:被检测出的不同严重程度故障对可靠性的影响显然不同,严重故障理应具有高优先权,这样,建模排错过程的队列系统应具备不同优先权,并考虑抢占式特征.同时,故障的严重程度又与故障间的关联性问题紧密联系;

(3) 当前,采用队列理论建模与处理排错过程的研究尚未有触及到排队论的核心,例如,在求解平均队列长度、平均逗留时间(平均等待时间+平均服务时间)、顾客到达服务机构需要等待的概率以及资源利用率(对应排错资源的使用情况)等.此外,排队过程本质上是一个生灭过程,对应故障被检测后进入队列,后又被排除掉.这样,应充分对多队列多通道(有限)排队系统的生率和灭率进行研究.

4.3 仿真技术——产生失效数据集与分析可靠性过程

解析模型与方法为求解上的方便,进行了大量的偏离真实测试与运行环境的假设,这使得所建立的模型存有很大偏差;相反,则会使得求解变得异常困难.另外,目前公开发表的失效数据集具有下面 3 个特点:① 数量有

限(作者搜集到 20 余个);② 数据集中数据数量也相对不多,一般不多于 21 周;③ 大多是 20 世纪 80 年代至本世纪初由知名公司发布.这些事实对 SRGM 的深入研究形成了障碍,使得新的模型难有验证的机会.

针对这两个方面的不足,软件可靠性仿真方法得到了研究人员的重视,并在近几年得到了深入发展.这样,仿真方法[109−111]成为一种获得数据集和进行可靠性过程分析的新尝试而被采用.

例如,基于离散事件的仿真[72−74]在可靠性分析上得到了广泛应用.

4.3.1 率控制的离散事件随机过程仿真

从观察失效计数的角度,软件测试过程是一种统计事件过程.在第 1.1 节 SRGM 基本假设之上,当假定在不相互重叠的时间内失效事件相互独立,且每一个时间步 dt 应充分小(即足够短),以确保最多只有一个失效发生,则可以采用率驱动事件过程 RDEP(rate-driven event processes)[112]的基础理论来进行处理.基于我们前期的研究工作[10,113],这里概要介绍 RDEP 理论.

设基于率的离散事件随机过程为$\{\beta_0(t),S_0,S_1,\ldots,S_k\}$,其中,事件 $\boldsymbol{E}$ 被定义为软件失效发生,$S_0,S_1,\ldots,S_k$ 为 $\boldsymbol{E}$ 的状态,$\beta_0(t)$即为 $\boldsymbol{E}$ 的率函数,这里也被称为冒险率,下标 0,1,k 表示时间 t 的离散取值.在给定 t 时刻,$\boldsymbol{E}$ 不发生的概率为

$$P_0(t)=\mathrm{e}^{-\lambda_0(0,t)} \tag{12}$$

其中,$\lambda_{t_0}(t_0,t)=\int_{t_0}^{t}\beta_0(\tau)\mathrm{d}\tau$.为确保状态非负以及确保事件最终必发生,下面两个约束必须得到满足:$\beta_0(t)\geq 0$ 和 $\lambda_0(0,\infty)=\infty$.由于 $\lambda_0(0,t)=\int_0^t\beta_0(\tau)\mathrm{d}\tau$,$\lambda_0(0,t)$被称为全部冒险率(函数).这样,首个软件失效事件发生时间 T 的概率分布函数和概率密度函数可表示为 $F_1(t)=1-P_0(t)$和 $f_1(t)=\beta_0(t)\mathrm{e}^{-\lambda_0(0,t)}$.利用连续型随机变量的数学期望公式,可知该失效发生的平均时间为

$$E(t)=\int_0^{\infty}t[\beta_0(t)\mathrm{e}^{-\lambda_0(0,t)}]\mathrm{d}t \tag{13}$$

下面的图 7 给出了率函数 ***beta***(t)作用下的单事件过程仿真.

```
Double single_event (double t, double dt, double (*beta)(double))
{ int event=0;
  While (event==0)
    { if (occurs(beta(t)*dt))
      event++;
      t+=dt; }
  return t; }
```

Fig.7 Single event process simulation of RDEP

图 7 RDEP 下单事件过程仿真

该程序仿真了失效事件发生在 t 时刻,并返回 t 值.其中,用以判断事件是否发生的 ***occurs***(***beta***(t)*dt)实现如下:通过随机函数 ***random***()产生[0,1)之间数,若 ***random***()<***beta***(t)*dt,则失效发生.此判断规则即是率函数控制下的随机事件仿真的实质所在.

4.3.2 考虑不完美排错环境的可靠性过程分析

不完美排错在 SRGM 研究体系中占有非常重要的地位,是 SRGM 走向深入研究的重要转折点.但由于其考虑随机过程因素更为细致,使得研究变得逐渐复杂.Chu[72]首次提出了不完美排错下软件测试过程的仿真过程:*PRO_IM_DEBUG*,与 Gokhale 所提出的构件软件可靠性分析仿真框架[69]是 RDEP 仿真方法自提出以来在 SRGM 研究上最为丰富的应用.*PRO_IM_DEBUG* 中包含了故障检测(detection)、排除(correction)、新故障引入与曝光(introduction+exposure)过程.上述仿真步骤实现的关键即为图 7 中给出的方法.*PRO_IM_DEBUG* 的最大优点是在队列模式下将不完美排错的典型过程进行仿真,开创了将真实测试过程中特征考虑到仿真中的先例.其不足是:虽考虑到排错的不彻底和新故障引入,但更为真实的不完美因素未被包含进来;虽考虑到排错人员的限制,但缺少 TE 等的考虑.

4.3.3 仿真方法的特殊用途

SRGM 的验证主要是度量所建立的模型与真实数据集之间的差异(包括拟合与预测,见第 5 节),因而若缺少

软件测试的失效数据集,则 SRGM 无从验证,这成为制约研究发展的主要障碍.由上面分析可知,RDEP 仿真能够产生[0,t]内被检测的、修复的、引入的故障个数,这与真实的失效数据集在本质上是相同的.为此,仿真方法促使了后来的研究人员充分认识到 RDEP 仿真的这一功用,将仿真结果与解析模型进行比较[114],或基于真实的数据集得到的参数估计结果[73],使不同仿真方法的比较成为可能.我们认为,仿真方法的这一独特之处,对于弥补当前失效数据集的不足势必会发挥更大的作用.

4.3.4 小 结

当前,仿真主要还是为测试过程管理提供参考:① 可以模拟软件测试过程,通过修改仿真参数来预判真实测试过程的基本特征;② 产生失效数据集,包括检测、修复、引入等故障数量,为某些 SRGM 进行验证提供数据支持.当前,仿真方法在研究 SRGM 中有下面几点需要考虑:

(1) 排错资源所涵盖的内容过于单一:当前,仅有的考虑到排错资源的仿真研究也仅仅将排错人员的个数作为资源,对排错人员的技能、排错策略与技术、用例选择、周期以及成本控制等尚没有考虑到;

(2) 在成本可控下实现高可靠性是软件可靠性工程的主要期望目标,仿真方法显然可以在管控成本方面发挥优势,但如何在仿真中实现测试成本的融入是关键问题.我们认为,在当前率控制下的仿真中,对故障的检测、分配、修复等环节应加入对 TE 的考虑,但要注意到下面的两点:① 不同操作的 TE 应有所区别;② TEF 应根据真实测试环境来确定,目前可用的 TEF 见表 4 中所列;

(3) 虽然仿真方法已得到研究人员的重视,但能够描述更为真实测试环境的仿真程序至今也尚未见诸文献,这极大制约了仿真方法的发展.同时,我们认为:应将注重抽样和模拟的(马尔科夫链)蒙克卡罗仿真方法融入进 RDEP 仿真方法中,二者的有机结合可实现将较为繁琐的解析模型借助仿真以更加细腻的方式来实现;

(4) 由于仿真自身的固有缺陷(例如,无法给出精确的失效时间和数值等),因而在应用仿真方法时要充分认识到其与工程实际的偏差.

4.4 小 结

传统的并延续至今的解析技术、基于排队论的分析技术以及以率(函数)作为驱动的离散事件为主的仿真技术已成为 SRGM 研究的 3 个关键技术,当前三者并存,表 5 对此进行了比较.

Table 5 Comparisons of key technologies

表 5 关键技术比较

技术类别	主要特征	优/缺点	适用环境
解析方法	① 借助数学方程来进行测试过程的建模,将故障检测、修复、引入等表示为若干个微分方程(组)形式; ② 需要通过求解得到的变量表达式或数值来分析测试过程中的现象; ③ 假设的不同使得数学建模偏差较大	优点:可将检测、修复等故障数量用数学表达式形式进行表示,便于与真实的数据集进行对比验证; 缺点:复杂的测试环境难以用数学方程进行建模,且求解的难度较大,有时只能借助数学工具给出数值解	解析方法的主要特定使得其在明确知晓变量间依赖关系时尤为适用,并通过求解获得定量的解析解或数值解,进而用以明确刻画测试中的部分子过程
排队论技术	① 将软件测试过程等效地看成排队系统; ② 被检测出的故障⇔顾客,软件失效率⇔顾客到达率,故障修复⇔服务窗口(服务人员),进而运用排队论的数学方法求解	优点:可以模拟测试过程中故障所经历的多个子过程,且支持故障类别的差异性等特点; 缺点:排队论的生灭过程理论使得当考虑测试过程中更多细节时变得异常复杂	排队论的结构模型特征,使其在面向测试过程中带有随机聚散、随机服务特点的系统时更具优势
仿真技术	① 用程序代码模拟软件测试的全过程或部分过程; ② 无需数学上的建模和求解,只需较少的参数设置即可驱动仿真; ③ 能够获得仿真数据集(检测、等待、修复、引入等)	优点:相比解析方法与排队论技术,不需要做过多的假设;通过参数设置的变化,可以及时得到仿真结果; 缺点:只能模拟并给出趋势性判断,不能确定精确的失效时间、各类型故障数量等	由于仿真技术只能给出趋势性的结果,因而其更适用于预知预判、辅助决策等场景

5 典型模型性能比较分析

本节基于多个公开发表的失效数据集,对选定的典型 SRGMs 进行性能验证,分析不同模型间的差异.

5.1 参与比较的SRGMs

SRGM 主要通过指数型(exponential)或者 *S*-shaped 模式[106]描述软件失效现象,且本质上二者均是基于 NHPP 的一种随机过程.30 几年来已由此衍生出多种 SRGM,我们经过梳理,将其分类归并为表 6 中的典型模型.

Table 6 Modeling comparisons of the SRGM

表 6 不同 SRGM 的建模比较

模型		类型	累积故障检测数量 $m(t)$(即平均值函数 *MVF*)与其他参变量设置解释
M-0: G-O		EX+PD (concave)	$m(t)=a[1-e^{-bt}]$;$a(t)=a$;$b(t)=b$.也被称为指数模型或 Musa 模型
S-shaped	M-1: Delayed *S*-shaped[35,36,67]	PD+ *S*-shaped	$m(t)=a[1-(1+bt)e^{-bt}]$;$a(t)=a$;$b(t)=b^2t/(1+bt)$是对 G-O 模型的改进
	M-2: Inflection *S*-shaped[49]	SS+PD+ Concave	$m(t)=\frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$;$a(t)=a$;$b(t)=\frac{b}{1+\beta e^{-bt}}$,如果$\beta$=0,得到 G-O 模型
Yamada	M-3: Yamada Exponential[49]	Concave+TEF	$m(t)=a[1-e^{-b\alpha(1-e^{-\beta t})}]$;$a(t)=a$;$b(t)=b\alpha\beta e^{-\beta t}$.尝试考虑到 TE
	M-4: Yamada Rayleigh[46]	*S*-shaped+TEF	$m(t)=a[1-e^{-b\alpha(1-e^{-\beta t^2/2})}]$;$a(t)=a$;$b(t)=b\alpha\beta te^{-\beta t^2/2}$.尝试考虑到 TE
	M-5: Yamada Weibull[38]	Concave+TEF	$m(t)=a[1-e^{-b\alpha(1-e^{-\beta t^{\delta}})}]$
Yamada Imperfect Debugging	M-6: Y-Exp[38]	Concave+ID	$m(t)=\frac{ab(e^{\alpha t}-e^{-bt})}{a+b}$;$a(t)=ae^{\alpha t}$,假定指数故障内容函数;$b(t)=b$
	M-7: Y-Lin[38]	Concave+ID	$m(t)=a(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha at$;$a(t)=a(1+\alpha t)$,$\alpha$为故障引入率;$b(t)=b$.
M-8: P-N-Z Model[96]		*S*-shaped+ Concave+ID	$m(t)=\frac{a\left[(1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha t\right]}{1+\beta e^{-bt}}$;$a(t)=a(1+\alpha t)$;$b(t)=\frac{b}{1+\beta e^{-bt}}$; $a(t)$为 t 的线性增函数;$b(t)$为弯曲 *S* 型函数,且非降
M-9: P-Z2[115]		ID	$m(t)=a(1+rt)(rt+e^{-rt}-1)$;$a(t)=a(1+rt)^2$
M-10: P-Z model[43]		*S*-shaped+ Concave+ID	$m(t)=\frac{(c+a)(1-e^{-bt})-\frac{ab}{b-\alpha}(e^{-\alpha t}-e^{-bt})}{1+\beta e^{-bt}}$;$a(t)=c+a(1-e^{-\alpha t})$;$b(t)=\frac{b}{1+\beta e^{-bt}}$
M-11: Zhang-Teng-Pham[116]		ID	$m(t)=\frac{a}{p-\beta}\left[\left(1-\frac{(1+\alpha)e^{-bt}}{1+\alpha e^{-bt}}\right)^{\frac{c}{b}(p-\beta)}\right]$;$a(t)=\beta(t)m(t)$,$\beta(t)=\beta$;$b(t)=\frac{c}{1+\alpha e^{-bt}}$
M-12: Pham Zhang IFD[95]		ID	$m(t)=a[1-e^{-bt}(1+(b+d)t+bdt^2)]$
M-13: P-Z[97]		ID	$m(t)=a(1+\beta t)\left[1-e^{-\frac{b^2t^2}{2(1+bt)}}\right]$;$a(t)=a(1+\beta t)$
M-14: Pham[96]		ID	$m(t)=\frac{a}{1+\sigma e^{-b(1+\sigma)t}}\left[(1-e^{-b(1+\sigma)t})\left(1-\frac{r}{b(1+\sigma)}\right)+rt\right]$;$a(t)=a(1+rt)$; $b(t)=\frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)t}}$,$\sigma$为测试过程学习因子,可正可负
M-15: Ohba-Chou[41]		ID	$m(t)=\frac{a}{1-r}[1-e^{-(1-r)bt}]$;$b(t)=b$;$a(t)=\frac{a}{1-r}[1-re^{-(1-r)bt}]$
M-16: Xie[92]		ID	$m(t)=b\left[\frac{a(1+kt)}{k-b}+\frac{r}{b+k}\left(\frac{(1+kt)^2}{2k}-\frac{(1+kt)^{1-\frac{b}{k}}-1}{k-b}\right)\right]$;$a(t)=a(1+rt)$, $p(t)=\frac{1}{1+kt}$,k 表示故障排除变化率
M-17: Huang[30]		TEF	$m(t)=\frac{a}{1-b}[1-e^{-bW^*(t)}]$;$W^*(t)=\frac{W}{1+Ae^{-\alpha t}}-\frac{W}{1+A}$;$a(t)=a$;$b(t)=b$
M-18: Yamada-TEF[117]		TEF	$m(t)=a[1-e^{-bW^*(t)}]$;$W^*(t)=W[1-e^{-bt^{\sigma}}]$ (此为 Weibull 类型 TEF)

Table 6 Modeling comparisons of the SRGM (Continued)

表 **6** 不同 SRGM 的建模比较(续)

模型	类型	累积故障检测数量 $m(t)$(即平均值函数 *MVF*)与其他参变量设置解释
M-19: Ahmad[54]	TEF	$m(t)=a\left[1-\mathrm{e}^{-bW(1-e^{-\beta t^2})^{\theta}}\right];W(t)=W\left[1-\mathrm{e}^{-\beta t^2}\right]^{\theta}$ (此为 Burr 类型 X TEF)
M-20: TEFIDM[6]	ID+TEF	$m(t)=\frac{a}{1-b}[1-\mathrm{e}^{-b(1-b)W^*(t)}];W^*(t)=\left(\frac{W}{1+A\mathrm{e}^{-\alpha t}}-\frac{W}{1+A}\right)$
M-21: SEWTEFIDM[5]	ID+TEF	$m(t)=\frac{a[1-\mathrm{e}^{-bW(t)}]}{1+[(1-\gamma)/\gamma]\mathrm{e}^{-bW(t)}};W(t)=W(1-\mathrm{e}^{-\varphi t^{\sigma}})^{\theta}$
M-22: SLTEFIDM[6]	ID+TEF	$m(t)=\frac{a}{1-r}\{1-[1+bW^*(t)]^{1-r}\mathrm{e}^{-b(1-r)W^*(t)}\};W^*(t)=\left(\frac{W}{1+A\mathrm{e}^{-\alpha t}}-\frac{W}{1+A}\right)$
M-23: WTECPM[31]	ID+TEF+CP	$m(t)=a[1-\mathrm{e}^{-b(W(t)-W(0))}];W(t)=\begin{cases}W(1-\mathrm{e}^{-\varphi_1 t^{\sigma_1}}), & 0\le t\le\tau\\ W(1-\mathrm{e}^{-\varphi_1\tau^{\sigma_1}}+\mathrm{e}^{-\varphi_2\tau^{\sigma_2}}-\mathrm{e}^{-\varphi_2 t^{\sigma_2}}), & t\ge\tau\end{cases}$
M-24: GLTECPM[25]	ID+TEF+CP	$m(t)=\begin{cases}a[1-\mathrm{e}^{-b_1(W(t)-W(0))}], & 0\le t\le\tau\\ a[1-\mathrm{e}^{-b_1[(W(\tau)-W(0))+b_2(W(t)-W(\tau))]}], & t>\tau\end{cases};W(t)=\frac{W}{\sqrt[\kappa]{1+A\mathrm{e}^{-\alpha\kappa t}}}$
M-25: ECPM[26]	ID+TEF+CP	$m(t)=\begin{cases}a\left[1-\mathrm{e}^{-\frac{1}{k+1}b_1(W(t)^{k+1}-W(0)^{k+1})}\right], & 0\le t\le\tau\\ a\left[1-\mathrm{e}^{-\frac{1}{k+1}[b_1(W(\tau)^{k+1}-W(0)^{k+1})+b_2(W(t)^{k+1}-W(\tau)^{k+1})]}\right], & t>\tau\end{cases},$ $W(t)$为 Rayleigh,Logistic,Weibull 这 3 种分布

注:类型包括:① EX:指数类型;② PD:完美排错模型;③ ID:不完美排错类型;④ TE:考虑到 TE;
⑤ CP:考虑到 CP;⑥ *S*-shaped:*S* 型;⑦ Concave:凹形

5.2 数据集选择及与模型验证的对应关系

现有 SRGM 的性能验证均是基于公开发表的失效数据集来进行的.我们搜集了 16 个相对完整的数据集,这些数据集可以分为两类:一类是含有故障检测时间 t 与累积检测到的故障数量 $m(t)$,另一类是还包括有测试工作量 $W(t)$.这里,由于需要比较的模型与数据集二者均比较多,为了更加合理地反映不同模型的性能差异,我们制定了下面的模型与数据集验证对应关系:(1) 在包含 TE 的数据集上,被比较的 SRGM 模型中应该涵盖对 TEF 的考虑;(2) 对不完美排错相关的 SRGM 的验证,要使得反应不完美排错的故障引入率 $\hat{r}$ 和故障修复概率 $\hat{p}$ 二者的拟合值存在且位于(0,1)之间.在我们前期研究工作[48,56]中,已对含有 TEF 的不完美排错模型进行了分析(对应 M-17~M-22);此外,考虑 CP 时,本质上只需分段研究即可(对应 M-23~M-25);另,由于页面限制(可联系作者,获得更多实验信息),这里只给出了部分模型在部分数据集上的验证结果及分析:M-6 至 M-16 在 DS_1[103],DS_2~DS_5[118–121],DS_6[61],DS_7[99],DS_8[97],DS_9[122]上验证.这些数据集主要来自于国际知名计算机公司或机构的大型计算机系统,可被描述为如下的三元组:$DS=\{t,N(t),W(t)\}$,其中:t 表示测试时间,通常以周为单位;$N(t)$为截止 t 时累积检测到的故障数量;$W(t)$为$[0,t]$内消耗的测试工作量 TE.失效数据集用于 SRGM 研究中进行模型的验证和性能评价,其相互之间有一定差异,表 7 给出了本文所用到的数据集情况.

Table 7 Failure data set

表 **7** 失效数据集描述

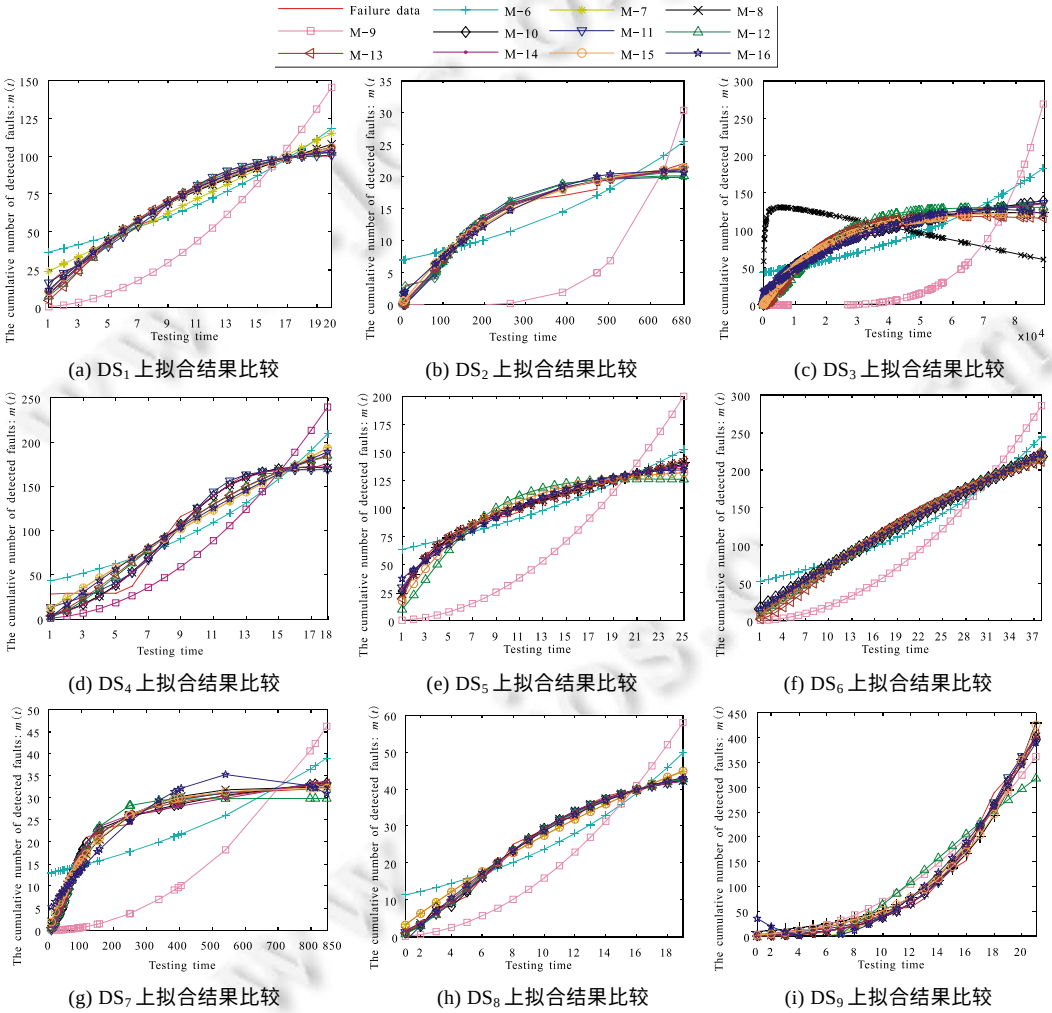
数据集	来源	记录时间	代码行数	测试中累计检测到的失效个数	累积消耗的测试工作量 TE
DS1[103]	Tandem(天腾)公司:计算机工程项目	记录 20 周	不明	16~100	519~10 000
DS2[118]	AT&T Bell 实验室:网络管理系统	记录 680.02 个 CPU 单位时间	不明	1~22	不明
DS3[119]	电信系统	1 个标准化时间	不明	0.05~1(标准化)	不明
DS4[120]	大型医疗记录系统	记录 18 周	内含 188 个软件构件	28~176	不明

Table 7 Failure data set (Continued)
表 7 失效数据集描述(续)

数据集	来源	记录时间	代码行数	测试中累计检测到的失效个数	累积消耗的测试工作量 TE
DS5[121]	Misra 系统	记录 25 小时	不明	27~136	不明
DS6[61]	Misra 系统	记录 38 周	1.5 百万行代码级	15~231	不明
DS7[99]	海军舰队计算机程序中心:海军战术数据系统	记录 849 天	含 38 个工程模块	1~34	不明
DS8[97]	Tandem Computer (天腾计算机)公司	记录 19 周	几百万行代码级别	1~42	不明
DS9[122]	航空程序	记录 21 周	9564 行 C 代码	2~403	不明

5.3 性能评价与分析

基于 SRGM 在数据集上的拟合结果,我们画出了 $m(t_i)$与真实失效数值 y_i 之间的拟合曲线,如图 8 所示.



(a) DS_1 上拟合结果比较 (b) DS_2 上拟合结果比较 (c) DS_3 上拟合结果比较

(d) DS_4 上拟合结果比较 (e) DS_5 上拟合结果比较 (f) DS_6 上拟合结果比较

(g) DS_7 上拟合结果比较 (h) DS_8 上拟合结果比较 (i) DS_9 上拟合结果比较

Fig.8 Fitting curves of M-6 to M-16 on DS_1~DS_9
图 8 M-6 至 M-16 在 DS_1~DS_9 上度量拟合曲线

模型的拟合曲线越接近于真实的失效数据曲线表明拟合效果越好,从图 8 可以看出:

(1) M-6~M-16 合计 11 个 SRGM 均是不完美排错模型,但在同一数据集上的性能差异表现迥异:整体上,

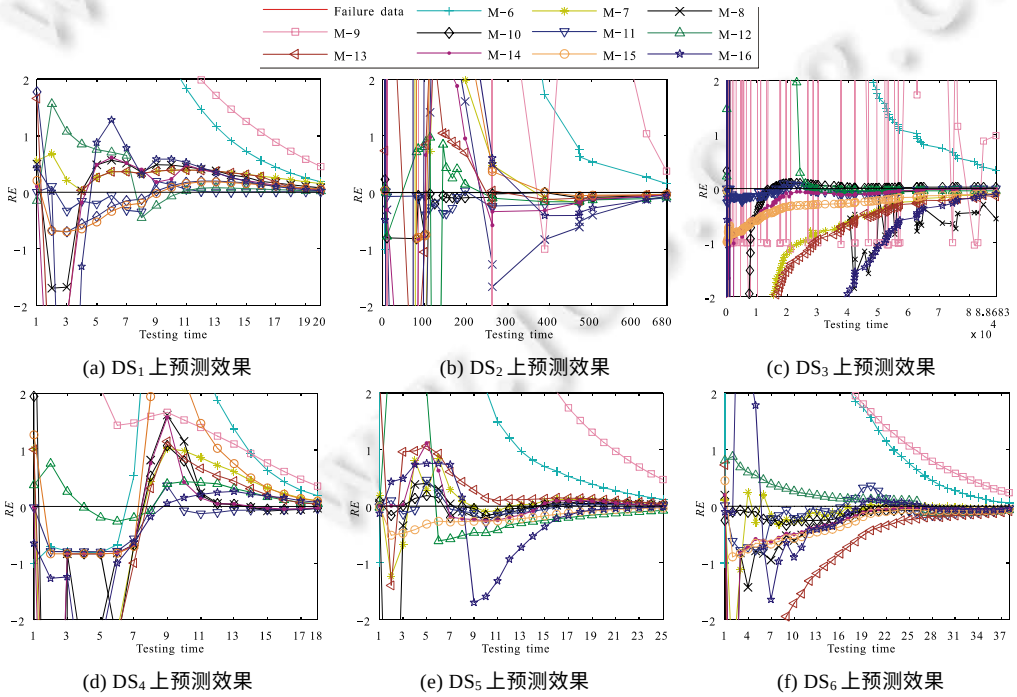
M-9 和 M-6 的拟合效果最差(如图 8(a)~8(g)所示,尤其是在图 8(c)中,不足表现尤为明显),其余模型的性能差别不是特别明显(除 M-8 在 DS_3 上的表现差强人意以外).在此 9 个数据集中,前 8 个呈现凸形状增长趋势,DS_9 为凹形状增长.这样,呈现凹形状的 M-9 和 M-6 模型与绝大多数为凸形状的数据集偏差较大;

(2) 在同一个数据集上,造成模型间差异的主要原因是所建立的微分方程(组)的不同(直接导致了所求解得到不同的 $m(t)$)以及对设定或求解 $a(t)$和 $b(t)$表达式的不同.以整体上表现较为优异的 M-11 为例,其建立的模型为 $\begin{cases} \dfrac{\mathrm{d}m(t)}{\mathrm{d}t} = b(t)[a(t) - pm(t)] \\ \dfrac{\mathrm{d}a(t)}{\mathrm{d}t} = \beta(t)\dfrac{\mathrm{d}m(t)}{\mathrm{d}t} \end{cases}$.该微分方程组的第 1 个子式中考虑到了排错的不完全性,第 2 个子式描述的是新故障引入情况.整体上,该模型是较为合理的,符合真实测试情况.另外,$b(t) = \dfrac{c}{1+\alpha \mathrm{e}^{-bt}}$ 为弯曲的 S 形变化,可适应多种真实测试环境;

(3) 由于考虑到 CP 的存在,使得 $m(t)$成为分段函数,充分兼顾到 CP 前后数据的结构变化,因而在含有 CP 的数据集上,考虑 CP 的模型通常要比不考虑 CP 的 SRGMs 表现得更为优秀;

(4) 为了进一步区分模型间的差异,需要计算其在 *MSE*,*RMS-PE*,*MEOP*,*Variation*,*TS*,*BMMRE*(这些标准值越小,表明拟合效果越好)和 *R-square*(*R-square* 越接近于 1 越好)比较标准[48,56]上的具体数值.这里,由于页面空间所限,没有给出不同模型在这些数据集上的比较标准值,可参照我们已取得的研究成果[48,56].一般而言,只有待比较的模型在这些标准值上存有绝对数量优势的大小关系时,才可判定模型间的优劣.因而对于模型性能比较问题,应根据用户对某些标准的偏好以及在明确的比较策略下,通过综合计算与比较分析才可给出定量的优劣排序,这也是 SRGM 亟待攻克的研究要点.

另一方面,在预测性能的验证上,我们画出了部分模型的 *RE* 曲线(*RE* 越快趋近于 0,表明预测效果越好),如图 9 所示.

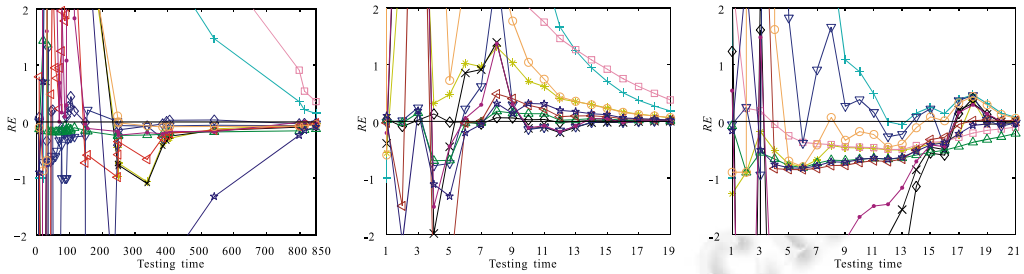


(a) DS_1 上预测效果 (b) DS_2 上预测效果 (c) DS_3 上预测效果

(d) DS_4 上预测效果 (e) DS_5 上预测效果 (f) DS_6 上预测效果

Fig.9 *RE* curves of M-6 to M-16 on DS_1~DS_9

图 9 M-6 至 M-16 在 DS_1~DS_9 上 *RE* 曲线



(g) DS_7 上预测效果 (h) DS_8 上预测效果 (i) DS_9 上预测效果

Fig.9 *RE* curves of M-6 to M-16 on DS_1~DS_9 (Continued)

图 9 M-6 至 M-16 在 DS_1~DS_9 上 *RE* 曲线(续)

预测曲线 *RE* 越快速接近于 0 标准线表明预测性能越好,从图 9 的预测曲线可以看出:

(1) 除在 DS_9 上 M-6 和 M-9 预测效果较好外(见图 9(i)),在其余 8 个数据集上,二者预测效果均排在其余 9 个模型之后.这可以解释为:二者所建立的数学模型偏离实际测试过程较严重,以及设定的 $b(t)$不具有任何变化特点所导致;

(2) 一个公共的现象是所有模型后期的预测性能要优于前期,尤其在测试时间跨过一半以后.例如,含有参数较多的模型 M-6,M-10,M-16 等在测试初期就抖动剧烈(如图 9(b)、图 9(g)~图 9(i)所示).这是因为预测性能本身需要一定量的数据作为支撑;同时,不同 SRGM 中参数较多,这使得只有数据集超过必要的大小后,预测性能才可以明显地表现出来;

(3) 在 DS_2 和 DS_9 上不同模型的起伏较大,表明预测性能并不稳定;但在 DS_1,DS_3,DS_5,DS_6 和 DS_8 上,参与比较的模型(除 M-6 和 M-9 以外)起伏较小,相对较为稳定.造成这种起伏较大现象的主要原因是:客观上,失效数据变化较大(例如存在 CP),以及模型并不具备良好的适应性而综合作用的结果.

5.4 讨 论

我们在 16 个公开发表的失效数据集上对表 5 中 26 个模型进行了大量的实验验证,综合前述实验结果与分析表明.

(1) 整体而言,模型的拟合与预测性能相一致:拟合性能优秀的模型在预测性能上也相对较为优秀,反之亦然.例如,M-6 和 M-9 仅在 DS_9 上的拟合与预测效果尚可(如图 8(i)和图 9(i)所示),在其余 8 个数据集上的拟合与预测均不理想.这可以被解释为:在稳定的测试环境下所记录的数据集不存在明显偏离其内在的增长趋势,这使得其与某些模型的本质正相吻合.另一方面,有的数据集使得部分模型的性能表现较好,却使得另外一些模型的性能差强人意;

(2) 不存在某个 SRGM 在所有数据集上表现优秀,也并非某个数据集适用于所有 SRGM.这可被解释为:不同数据集来源于不同公司的不同类型软件系统测试过程,因而数据集内在直接的本质差异使得其不可能适用于所有 SRGM;

(3) 因此,至今也不存在某个 SRGM 适用于所有的测试场合,这是由下面两个原因所引发:

- 客观上,不同失效数据集来源于不同公司在不同测试环境与策略下采用不尽相同的故障收集机制所获得的,因而,某特定数据集只能与有限个数的 SRGM 相适应;
- 主观上,科研人员对故障检测至排除过程的认识与数学建模差异巨大,这直接造成求解得到的累积检测的故障数量 $m(t)$千差万别,进而使得可靠性 $R(t)$差异较大.但在某特定测试环境下,存在某个 SRGM 比其他的在拟合与预测性能上要好,这从图 8、图 9 以及上面的分析中可以得到;

(4) 如何采取合适的算法对 SRGM 性能进行有效评价也是值得研究的问题:

- 实际应用中,一种行之有效的方法是将多种 SRGM 进行混合应用,以求得整体上的最优效果;
- 这需要建立已有模型与数据集之间的对应关系,或者针对某数据集,为其寻找合适的 SRGM.为

此,如何制定出合适的综合性评价标准,在给定数据集上选出优秀的 SRGM,已成为今后研究的重要方向.

6 面临的新挑战、研究趋势和亟待解决的问题

SRGM 发展至今,建立度量与预测准确的模型仍是当前和今后 SRGM 发展研究中的主要目标,在实现该目标过程中会出现诸多挑战和亟待解决的问题,需要预判研究的趋势所在.这里,我们进行此方面的分析.

6.1 面临的新挑战

(1) 不完美排错建模挑战

测试过程受各种随机因素的制约,使得故障的检测至修复过程呈现出不完美的特点.

不完美排错[12,15,23,24,33,63,64]不仅局限在排错的不完全性或引入新故障上,其实质是真实的软件开发与测试环境中随机扰动因素的总称.基于我们前期的实验和研究成果[48,56]以及随着当前对不完美排错认识的深入进展[85,97−100],能够将测试环境的随机性与复杂性通过合适的数学进行建模描述,正在成为 SRGM 研究的新挑战.

(2) 复杂模型求解与验证的挑战

从更为全面细致的视角考虑真实测试环境,SRGM 中融入更多的影响因素能够使得所建立的模型更为细腻.这是当前研究中 SRGM 越发复杂的主要原因所在,但至少存在下面两个问题需要兼顾:

- 不断上升的模型复杂度直接导致了解析求解的难度增加,有些只能借助数学求解软件来给出近似结果.但即便借助随机过程中的理论也使得模型较为复杂,很多情况下只能给出上下限[123],甚至存在已无法求解和分析的情况;
- 通常,复杂模型在拟合能力上表现得较为优秀,在预测初期并不理想.为此,复杂模型验证时需要比简单模型验证所需更多的数据量,这是复杂模型不可回避的事实.

如何既兼顾实际测试过程又使得在数学处理上可接受,成为 SRGM 研究发展进程中需要破解的难题.

(3) 由单一可靠性向可信性过渡的挑战

基于文献[124]中的软件生命周期划分,从图 10 可以看出:测试阶段是软件开发的最后一个子过程,采用 SRGM 来评估软件制品的可靠性.当前研究中,开发测试阶段主要关注可靠性,从 Fault→可靠性角度应用 SRGM 进行可靠性分析;操作运行阶段主要关注可信性,从 Attack→可信性的角度采用各种技术进行可信性评估与防范攻击.

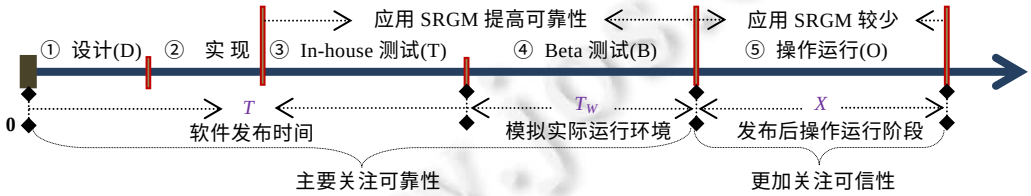


Fig.10 SRGM in the life cycle of software and evolution from reliability to trustworthiness

图 10 软件生命周期内应用 SRGM 情况及可靠性到可信性的变迁

目前,SRGM 仅从提高可靠性的角度来研究,能够从涵盖可靠性与安全性(包括可用性、机密性与完整性)的角度来研究,是应该值得关注的,因为作为一个集合属性,可信性(可靠性+安全性)[125,126]已越来越引起重视.

综上,如何破解这几个方面带来的挑战是 SRGM 研究中需要解决的基础性理论问题.

6.2 研究趋势概要分析

(1) SRGM 研究向非参量方法转移

目前,占据主要地位的解析方法主要采用统计推断技术,记录失效数据,采用数学建模的方法度量和预测失效及可靠性,这种基于参量方法的缺陷表现在:① 为便于建立和求解数学模型,往往进行过度假设[24,32,33,63],使

得研究与实际测试及运行情况偏差过大;② 当状态空间过大时,会遇到状态空间爆炸问题[85,127],使得模型难以进行定量解析,停滞不前.从属于机器学习范畴的非参量方法[72],包括支持向量机、遗传规划(算法)、人工神经网络等建立的启发式模型可尝试用于解决上述问题(这些方法尚未被真正引入到 SRGM 的研究中).近年来,利用(实数)遗传算法[128,129]或群智能优化算法[130]对 SRGM 进行参数估计、优化和最优发布时间的研究已开始出现.应用非参量方法时,必须要将测试/运行时间融入到模型中,以便使得能够研究其随着时间的进行引发各种特征的演变效应.随着解析方法的弊端日益显露,这些方法可成为 SRGM 研究的有效替代方法.

(2) 失效数据集与考虑软件结构特征相结合下的 SRGM 研究

SRGM 主要面向规模较大的软件系统开发与测试阶段的可靠性提高,对失效数据的收集是由专业测试人员长时间内完成的,获得的失效数据集往往具有较大的规模体量.当前研究中,对失效数据集的利用并不充分,仅仅单纯地用于参数估计与评估[55,63,64],失效数据集包含内容的不足也限制了 SRGM 研究的深入.传统的 SRGM 研究完全依赖于失效数据集进行拟合与预测分析,但软件规模、复杂度的上升以及开发技术的进步,软件自身结构特征因素尚未在研究中发挥作用.将失效数据集以及软件结构特征相结合考虑,把握二者的融合特征,理应成为推动 SRGM 深入研究的动力.

(3) 仿真研究应向更加靠近实际测试过程过渡

严格来看,率控制下的离散事件仿真在 SRGM 研究中已得到重视,但目前仅能对故障的检测、修复、引入进行仿真,尚没有触及真实软件测试过程中更多的排错细节.此外,如何将测试资源的消耗进行协调性仿真,在当前研究中也被忽略,这使得孤立地从故障排除角度来仿真可靠性过程缺乏必要的工程实际背景.如何考虑到更多的测试子过程,是仿真技术向前发展的必然趋势.

(4) 队列理论支配下 SRGM 的深入研究

当前,应用队列技术在 SRGM 的研究中多是对故障修复与排除进行多通道建模,对故障的检测过程则未采用队列模型,这与实际大型软件开发中多个测试小组的并行检测事实不相符.队列理论是较为成熟的一种随机过程,因而其更多的理论内容可以被引申至 SRGM 研究中,但目前,这方面研究工作还未有效展开.另外,仿真的非精准性与队列的定量性,以及二者都需要考虑更多真实的测试细节目标,使得二者应深入结合起来,共同使得 SRGM 的研究更加深入.

(5) 考虑“成本-可靠性”的 SRGM 评价

在软件开发过程中,管理人员若应用 SRGM 来估计可靠性的增长,可选择多个 SRGM 来并行应用[6].第 5 节的分析已指出,并不存在一个 SRGM 能对整个测试过程和环境下给出全程最优的结果.这样,在选择 SRGM 上,我们认为,仅凭借可靠性 $R(t)$的高低[131,132]并不足以判别其优劣.因为可靠性的提高是以测试资源的消耗和成本支出为代价的,这样必须综合以成本-可靠性作为评价的标准.在具体的评价方法上,可运用多属性决策或各种度量偏差距离的算法来评价不同 SRGM 的性能.

6.3 需要亟待解决的问题

(1) 时间因素——非参量求解映射进 SRGM 研究的关键

可靠性增长模型不同于可靠性评测的一个最显著特征是,将时间(测试时间或发布后的执行时间)作为一个重要的参变量.例如,$R(t)$是时间 t 的函数,表征随着时间的变化情况;而可靠性评测的结果多为某时刻的可靠性值 R.这样,从机器学习等非线性方法用于解决 SRGM 的角度出发,就必须要把时间因素有效地融入进去.当前,基于非参量的求解方法(支持向量机、神经网络、(李群)机器学习等)在可靠性评测中已开始得到尝试,但要将其引入到 SRGM 中,就必须要解决如何将时间因素考虑进去,这是亟待解决的问题之一.

(2) 新数据集——制约 SRGM 发展的瓶颈

SRGM 性能验证必须借助于失效数据集,但进入 21 世纪以来,新的数据集以及针对各种新软件形式的失效数据集迟迟未发布,这已成为制约 SRGM 深入向前发展的严重障碍.虽然仿真方法能够产生失效数据集,但仿真进行了假设,与真实的测试环境存有较大不同.此外,为了增大失效数据量,应丰富失效数据的收集记录方式,现有多是以周为单位累积记录失效个数,应支持以日历时间记录失效个数.

7 结束语

综上可以看出:SRGM 的发展经历了早期以 G-O 模型为代表的经典完美阶段到考虑 TE,CP 并融入更多实际因素的不完美阶段;从注重数学建模的定量解析方法到逐步引入仿真技术;由被动的基于发布的失效集进行验证到基于仿真的方法模拟测试过程来获取失效数据;自黑盒软件形式到构件软件形态;由强调从累积检测的故障数量研究可靠性的动态提高到扩展至测试资源管控和软件发布策略范畴.这些发展与变迁,促使 SRGM 研究呈现出丰富多样的内涵与视角,促进了软件可靠性研究得到深入发展.

本文对 SRGM 研究进行了全面综述,包括其建模流程、功用、影响因素分析、框架式建模与模型分类、技术类别分类讨论以及典型模型比较分析,并对制约 SRGM 发展的问题、挑战与未来趋势进行了分析,期望通过我们的这些工作,能为科研人员提供有益的借鉴参考,并为 SRGM 的进一步发展做出贡献.

致谢 在此,我们向本文参考文献中研究人员所做的大量基础工作表示真诚的感谢!对本文在写作与完善工作过程中给予无私支持和提供宝贵建议、意见的同行致谢!特别感谢审稿人,他们提出的宝贵意见和建议对于本文整体水平的提高有很大帮助!

References:

[1] Okamura H, Dohi T. A novel framework of software reliability evaluation with software reliability growth models and software metrics. In: Guerrero JE, ed. Proc. of the IEEE 15th Int'l Symp. on High-Assurance Systems Engineering (HASE). Miami Beach: IEEE, 2014. 97–104. [doi: 10.1109/HASE.2014.22]

[2] Yamada S. Software Reliability Modeling: Fundamentals and Applications. Tokyo: Springer-Verlag, 2014. 1–38. [doi: 10.1007/978-4-431-54565-1]

[3] Farooq SU, Quadri SMK, Ahmad N. Metrics, models and measurements in software reliability. In: Szakál A, ed. Proc. of the IEEE 10th Int'l Symp. on Applied Machine Intelligence and Informatics (SAMI). Herl'any: IEEE, 2012. 441–449. [doi: 10.1109/SAMI.2012.6209008]

[4] Okamura H, Etani Y, Dohi T. Quantifying the effectiveness of testing efforts on software fault detection with a logit software reliability growth model. In: O'Conner L, ed. Proc. of the 2011 Joint Conf. of the 21st Int'l Workshop on Software Measurement and the 6th Int'l Conf. on Software Process and Product Measurement (IWSM-MENSURA). Nara: IEEE, 2011. 62–68. [doi: 10.1109/IWSM-MENSURA.2011.26]

[5] Ahmad N, Khan MGM, Rafi LS. A study of testing-effort dependent inflection *s*-shaped software reliability growth models with imperfect debugging. Int'l Journal of Quality & Reliability Management, 2010,27(1):89–110. [doi: 10.1108/02656711011009335]

[6] Huang CY, Kuo SY, Lyu MR. An assessment of testing-effort dependent software reliability growth models. IEEE Trans. on Reliability, 2007,56(2):198–211. [doi: 10.1109/TR.2007.895301]

[7] Bokhari MU, Ahmad N. Incorporating burr type XII testing-efforts into software reliability growth modeling and actual data analysis with applications. Journal of Software, 2014,9(6):1389–1400. [doi: 10.4304/jsw.9.6.1389-1400]

[8] Lee TQ, Yeh CW, Fang CC. Bayesian software reliability prediction based on yamada delayed s-shaped model. Applied Mechanics and Materials, 2014,490:1267–1278. [doi: 10.4028/www.scientific.net/AMM.490-491.1267]

[9] Zhang C, Cui G, Liu HW, Meng FC, Fu ZC. Software test resources and cost control and optimal release policy. Journal of Harbin Institute of Technology, 2014,46(5):51–58 (in Chinese with English abstract). [doi: 10.11918/j.issn.0367-6234.2014.05.009]

[10] Zhang C, Cui G, Liu HW, Meng FC. Component-Based software reliability process technologies. Chinese Journal of Computers, 2014,37(12):2586–2612 (in Chinese with English abstract). http://www.cnki.net/kcms/detail/11.1826.TP.20140828.1915.003.html

[11] Quadri SMK, Ahmad N, Farooq SU. Software reliability growth modeling with generalized exponential testing-effort and optimal software release policy. Global Journal of Computer Science and Technology, 2011,11(2):27–42.

[12] Manjula T, Jain M, Gulati TR. Cost optimization of a software reliability growth model with imperfect debugging and a fault reduction factor. ANZIAM Journal, 2014,55:C182–C196. [doi: 10.21914/anziamj.v55i0.7834]

[13] Princy BA, Sridhar S. Measuring software reliability and release time using SRGM tool. Int'l Journal of Scientific Research and Education, 2014,2(5):785−796.

[14] Kapur PK, Aggarwal AG, Tandon A. Two dimensional software reliability growth model with faults of different severity. Communications in Dependability and Quality Management, 2010,13(3):98−110.

[15] Kapur PK, Singh O, Shrivastava AK. Optimal price and testing time of a software under warranty and two types of imperfect debugging. Int'l Journal of System Assurance Engineering and Management, 2014,5(2):120−126. [doi: 10.1007/s13198-014-0221-x]

[16] Li X, Xie M, Ng SH. Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points. Applied Mathematical Modelling, 2010,34(11):3560−3570. [doi: 10.1016/j.apm.2010.03.006]

[17] Hsu CJ, Huang CY, Chang JR. Enhancing software reliability modeling and prediction through the introduction of time-variable fault reduction factor. Applied Mathematical Modelling, 2011,35(1):506−521. [doi: 10.1016/j.apm.2010.07.017]

[18] Huang CY, Hung TY. Software reliability analysis and assessment using queueing models with multiple change-points. Computers & Mathematics with Applications, 2010,60(7):2015−2030. [doi: 10.1016/j.camwa.2010.07.039]

[19] Almering V, van Genuchten M, Cloudt G, Sonnemans PJM. Using software reliability growth models in practice. Software, 2007,24(6):82−88. [doi: 10.1109/MS.2007.182]

[20] Kapur PK, Goswami DN, Gupta A. A software reliability growth model with testing effort dependent learning function for distributed systems. Int'l Journal of Reliability, Quality and Safety Engineering, 2004,11(04):365−377. [doi: 10.1142/S0218539304001579]

[21] Huang CY, Lyu MR. Estimation and analysis of some generalized multiple change-point software reliability models. IEEE Trans. on Reliability, 2011,60(2):498−514. [doi: 10.1109/TR.2011.2134350]

[22] Goel L, Okumoto K. Time-Dependent error-detection rate model for software reliability and other performance measures. IEEE Trans. on Reliability, 1979,R-28(3):206−211. [doi: 10.1109/TR.1979.5220566]

[23] Peng R, Li YF, Zhang WJ, Hu QP. Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction. Reliability Engineering & System Safety, 2014,126:37−43. [doi: 10.1016/j.ress.2014.01.004]

[24] Kapur PK, Pham H, Anand S, Yadav K. A unified approach for developing software reliability growth models in the presence of imperfect debugging and error generation. IEEE Trans. on Reliability, 2011,60(1):331−340. [doi: 10.1109/TR.2010.2103590]

[25] Huang CY. Performance analysis of software reliability growth models with testing-effort and change-point. Journal of Systems and Software, 2005,76(2):181−194. [doi: 10.1016/j.jss.2004.04.024]

[26] Kapur PK, Singh VB, Anand S, Yadavalli VSS. Software reliability growth model with change-point and effort control using a power function of the testing time. Int'l Journal of Production Research, 2008,46(3):771−787. [doi: 10.1080/00207540600926113]

[27] Huang CY. Cost-Reliability-Optimal release policy for software reliability models incorporating improvements in testing efficiency. Journal of Systems and Software, 2005,77(2):139−155. [doi: 10.1016/j.jss.2004.10.014]

[28] Huang CY, Lo JH. Optimal resource allocation for cost and reliability of modular software systems in the testing phase. The Journal of Systems and Software, 2006,79(5):653−664. [doi: 10.1016/j.jss.2005.06.039]

[29] Kuo SY, Huang CY, Lyu MR. Framework for modeling software reliability, using various testing-efforts and fault-detection rates. IEEE Trans. on Reliability, 2001,50(3):310−320. [doi: 10.1109/24.974129]

[30] Huang CY, Kuo SY. Analysis of incorporating logistic testing-effort function into software reliability modeling. IEEE Trans. on Reliability, 2002,51(3):261−270. [doi: 10.1109/TR.2002.801847]

[31] Lin CT, Huang CY. Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models. Journal of Systems and Software, 2008,81(6):1025−1038. [doi: 10.1016/j.jss.2007.10.002]

[32] Singh O, Garmabaki AHS, Kapur PK. Unified framework for developing two dimensional software reliability growth models with change point. In: Proc. of the IEEE Int'l Conf. on Quality and Reliability (ICQR). Bangkok: IEEE, 2011. 570−574. [doi: 10.1109/ICQR.2011.6031604]

[33] Roy P, Mahapatra GS, Dey KN. An *S*-shaped software reliability model with imperfect debugging and improved testing learning process. Int'l Journal of Reliability and Safety, 2013,7(4):372−387. [doi: 10.1504/IJRS.2013.057423]

[34] Pham H. Loglog fault-detection rate and testing coverage software reliability models subject to random environments. Vietnam Journal of Computer Science, 2014,1(1):39−45. [doi: 10.1007/s40595-013-0003-4]

[35] Yamada S, Ohba M, Osaki S. *S*-Shaped software reliability growth models and their applications. IEEE Trans. on Reliability, 1984, 33(4):289−292. [doi: 10.1109/TR.1984.5221826]

[36] Yamada S, Ohba M, Osaki S. *S*-Shaped reliability growth modeling for software error detection. IEEE Trans. on Reliability, 1983, 32(5):475−484. [doi: 10.1109/TR.1983.5221735]

[37] Ohba M. Stochastic Models in Reliability Theory. Berlin, Heidelberg: Springer-Verlag, 1984. 144−162. [doi: 10.1007/978-3-642-45587-2]

[38] Yamada S, Tokuno K, Osaki S. Imperfect debugging models with fault introduction rate for software reliability assessment. Int'l Journal of Systems Science, 1992,23(12):2241−2252. [doi: 10.1080/00207729208949452]

[39] Wang JY, Wu ZB, Shu YJ, Zhang Z. Software reliability model with irregular changes of fault detection rate. Ruan Jian Xue Bao/ Journal of Software, 2015,26(10):2465−2484 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4746.htm [doi: 10.13328/j.cnki.jos.004746]

[40] Goel AL. Software reliability models: Assumptions, limitations, and applicability. IEEE Trans. on Software Engineering, 1985,12: 1411−1423. [doi: 10.1109/TSE.1985.232177]

[41] Ohba M, Chou XM. Does imperfect debugging affect software reliability growth? In: Proc. of the 11th Int'l Conf. on Software Engineering. Pittsburgh: ACM Press, 1989. 237−244. [doi: 10.1145/74587.74619]

[42] Lo JH, Huang CY. Incorporating imperfect debugging into software fault processes. In: Proc. of the 10th IEEE Region Conf. on TENCON. Chiang Mai: IEEE, 2004. 326−329. [doi: 10.1109/TENCON.2004.1414597]

[43] Pham H, Zhang X. An NHPP software reliability model and its comparison. Int'l Journal of Reliability, Quality and Safety Engineering, 1997,4(3):269−282. [doi: 10.1142/S0218539397000199]

[44] Schneidewind NF. Modelling the fault correction process. In: Proc. of the 12th Int'l Symp. on Software Reliability Engineering. IEEE, 2001. 185−190. [doi: 10.1109/ISSRE.2001.989472]

[45] Schneidewind NF. Fault correction profiles. In: Proc. of the 14th Int'l Symp. on Software Reliability Engineering. IEEE, 2003. 257−267. [doi: 10.1109/ISSRE.2003.1251048]

[46] Singh O, Kapur R, Singh J. Considering the effect of learning with two types of imperfect debugging in software reliability growth modeling. Communications in Dependability and Quality Management, 2010,13(4):29−39.

[47] Khatri SK, Kapur PK, Johri P. Flexible discrete software reliability growth model for distributed environment incorporating two types of imperfect debugging. In: Guerrero JE, ed. Proc. of the IEEE 2nd Int'l Conf. on Advanced Computing & Communication Technologies (ACCT). Rohtak: IEEE, 2012. 57−63. [doi: 10.1109/ACCT.2012.54]

[48] Zhang C, Cui G, Liu HW, Meng FC, Wu SX. A unified and flexible framework of imperfect debugging dependent SRGMs with testing-effort. Journal of Multimedia, 2014,9(2):310−317. [doi: 10.4304/jmm.9.2.310-317]

[49] Yamada S, Ohtera H, Narihisa H. Software reliability growth models with testing-effort. IEEE Trans. on Reliability, 1986,35(1): 19−23. [doi: 10.1109/TR.1986.4335332]

[50] Huang CY, Kuo SY, Chen Y. Analysis of a software reliability growth model with logistic testing-effort function. In: Werner B, ed. Proc. of the IEEE 8th Int'l Symp. on Software Reliability Engineering. Albuquerque: IEEE, 1997. 378−388. [doi: 10.1109/ISSRE.1997.630886]

[51] Huang CY, Lo JH, Kuo SY, Lyu MR. Software reliability modeling and cost estimation incorporating testing-effort and efficiency. In: Titsworth M, ed. Proc. of the IEEE 10th Int'l Symp. on Software Reliability Engineering. 1999. 62−72. [doi: 10.1109/ISSRE.1999.809311]

[52] Huang CY, Kuo SY, Lyu MR. Effort-Index-Based software reliability growth models and performance assessment. In: Rawlinson A, ed. Proc. of the IEEE 24th Annual Int'l Conf. on Computer Software and Applications (COMPSAC). Taipei: IEEE, 2000. 454−459. [doi: 10.1109/CMPSAC.2000.884764]

[53] Ahmad N, Bokhari MU, Quadri S MK, Khan MG. The exponentiated weibull software reliability growth model with various testing-efforts and optimal release policy. Int'l Journal of Quality & Reliability Management, 2008,25(2):211−235. [doi: 10.1108/02656710810846952]

[54] Ahmad N, Khan MGM, Quadri SMK, Kumar M. Modelling and analysis of software reliability with Burr type X testing-effort and release-time determination. Journal of Modelling in Management, 2009,4(1):28−54. [doi: 10.1108/17465660910943748]

[55] Ahmad N, Khan MGM, Rafi LS. Analysis of an inflection *S*-shaped software reliability model considering log-logistic testing-effort and imperfect debugging. Int'l Journal of Computer Science and Network Security, 2011,11(1):161−171.

[56] Zhang C, Cui G, Meng FC, Liu HW, Wu SX. A study of optimal release policy for SRGM with imperfect debugging. Journal of Engineering Science and Technology Review, 2013,6(3):111−118.

[57] Li Q, Li H, Lu M. Incorporating S-shaped testing-effort functions into NHPP software reliability model with imperfect debugging. Journal of Systems Engineering and Electronics, 2015,26(1):190−207. [doi: 10.1109/JSEE.2015.00024]

[58] Chang YP. Estimation of parameters for non-homogeneous Poisson process: Software reliability with change-point model. Communications in Statistics-Simulation and Computation, 2001,30(3):623−635. [doi: 10.1081/SAC-100105083]

[59] Wang Z, Wang J. Parameter estimation of some NHPP software reliability models with change-point. Communications in Statistics-Simulation and Computation, 2005,34(1):121−134. [doi: 10.1081/SAC-200047098]

[60] Zou FZ. A change-point perspective on the software failure process. Software Testing, Verification and Reliability, 2003,13(2): 85−93. [doi: 10.1002/stvr.268]

[61] Shyur HJ. A stochastic software reliability model with imperfect-debugging and change-point. Journal of Systems and Software, 2003,66(2):135−141. [doi: 10.1016/S0164-1212(02)00071-7]

[62] Achcar JA, Rodrigues ER, Paulino CD, Soares P. Non-Homogeneous Poisson models with a change-point: an application to ozone peaks in Mexico city. Environmental and Ecological Statistics, 2010,17(4):521−541. [doi: 10.1007/s10651-009-0114-3]

[63] Jain M, Manjula T, Gulati TR. Prediction of reliability growth and warranty cost of software with fault reduction factor, imperfect debugging and multiple change point. Int'l Journal of Operational Research, 2014,21(2):201−220. [doi: 10.1504/IJOR.2014.064544]

[64] Jain M, Manjula T, Gulati TR. Imperfect debugging study of SRGM with fault reduction factor and multiple change point. Int'l Journal of Mathematics in Operational Research, 2014,6(2):155−175. [doi: 10.1504/IJMOR.2014.059526]

[65] Langberg N, Singpurwalla ND. A unification of some software reliability models. SIAM J. Scientific and Statistical Computing, 1985,6(3):781−790. [doi: 10.1137/0906053]

[66] Chen Y, Singpurwalla ND. Unification of software reliability models by self-exciting point processes. Advances in Applied Probability, 1997,29:337−352. [doi: 10.2307/1428006]

[67] Huang CY, Lyu MR, Kuo SY. A unified scheme of some nonhomogenous Poisson process models for software reliability estimation. IEEE Trans. on Software Engineering, 2003,29(3):261−269. [doi: 10.1109/TSE.2003.1183936]

[68] Wu YP, Hu QP, Xie M, Ng SH. Modeling and analysis of software fault detection and correction process by considering time dependency. IEEE Trans. on Reliability, 2007,56(4):629−642. [doi: 10.1109/TR.2007.909760]

[69] Kapur PK, Anand S, Inoue S, Yamada S. A unified approach for developing software reliability growth model using infinite server queuing model. Int'l Journal of Reliability, Quality and Safety Engineering, 2010,17(5):401−424. [doi: 10.1142/S0218539310003871]

[70] Lo JH. Toward a unified approach to software reliability modeling under imperfect debugging. Applied Mechanics & Materials, 2015. 979−982. [doi: 10.4028/www.scientific.net/AMM.764-765.979]

[71] Tausworthe RC, Lyu MR. A generalized technique for simulating software reliability. IEEE Software, 1996,13(2):77−88 [doi: 10.1109/52.506464]

[72] Chu TL. Analyzing the effect of imperfect debugging on software fault detection on software fault detection and correction processes via a simulation work. Mathematical and Computer Modelling, 2011,54:3046−3064. [doi: 10.1016/j.mcm.2011.07.033]

[73] Hou CY, Chen C, Wang JS, Lin S. Software reliability process simulation considering imperfect debugging. High Technology Letter, 2014,24(6):558−564 (in Chinese with English abstract). [doi: 10.3772/j.issn.1002-0470.2014.06.002]

[74] Gokhale SS, Lyu MR, Trivedi KS. Incorporating fault debugging activities into software reliability models: A simulation approach. IEEE Trans. on Reliability, 2006,55(2):281−292. [doi: 10.1109/TR.2006.874911]

[75] Lin CT, Huang CY. Staffing level and cost analyses for software debugging activities through rate-based simulation approaches. IEEE Trans. on Reliability, 2009,58(4):711−724. [doi: 10.1109/TR.2009.2019669]

[76] Dohi T, Matsuoka T, Osaki S. An infinite server queuing model for assessment of the software reliability. Electronics and Communications in Japan (Part III: Fundamental Electronic Science), 2002,85(3):43−51. [doi: 10.1002/ecjc.1078]

[77] Dohi T, Osaki S, Trivedi KS. An infinite server queueing approach for describing software reliability growth: Unified modeling and estimation framework. In: Werner B, ed. Proc. of the IEEE 11th Asia-Pacific Conf. on Software Engineering. Busan: IEEE, 2004. 110−119. [doi: 10.1109/APSEC.2004.29]

[78] Huang CY, Huang WC. Software reliability analysis and measurement using finite and infinite server queueing models. IEEE Trans. on Reliability, 2008,57(1):192−203. [doi: 10.1109/TR.2007.909777]

[79] Huang CY, Hung TY, Hsu CJ. Software reliability prediction and analysis using queueing models with multiple change-points. In: Ceballos S, ed. Proc. of the IEEE 3rd Int'l Conf. on Secure Software Integration and Reliability Improvement (SSIRI). Shanghai: IEEE, 2009. 212−221. [doi: 10.1109/SSIRI.2009.11]

[80] Okumoto K, Goel AL. Optimum release time for software systems based on reliability and cost criteria. Journal of Systems and Software, 1980,1:315−318. [doi: 10.1016/0164-1212(79)90033-5]

[81] Huang CY, Lyu MR. Optimal release time for software systems considering cost, testing-effort, and test efficiency. IEEE Trans. on Reliability, 2005,54(4):583−591. [doi: 10.1109/TR.2005.859230]

[82] Yang B, Hu H, Jia L. A study of uncertainty in software cost and its impact on optimal software release time. IEEE Trans. on Software Engineering, 2008,34(6):813−825. [doi: 10.1109/TSE.2008.47]

[83] Manjula T, Jain M, Gulati TR. Optimal release policies of delayed discrete reliability growth model with imperfect debugging. Journal of Testing & Evaluation, 2015. [doi: 10.1520/JTE20150043]

[84] Yang J, Liu Y, Xie M, Zhao M. Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes. Journal of Systems and Software, 2016,115:102−110. [doi: 10.1016/j.jss.2016.01.025]

[85] Cao P, Dong Z, Liu K, Cai KY. Quantitative effects of software testing on reliability improvement in the presence of imperfect debugging. Information Sciences, 2012,218(2013):119−132. [doi: 10.1016/j.ins.2012.06.034]

[86] Anniprincy B, Sridhar S. Prediction of software reliability using COBB-DOUGLAS model in SRGM. Journal of Theoretical and Applied Information Technology, 2014,62(2):355−363.

[87] Li HF, Wang SQ, Liu C, Zheng J, Li Z. Software reliability model considering both testing effort and testing coverage. Ruan Jian Xue Bao/Journal of Software, 2013,24(4):749−760 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4527.htm [doi: 10.3724/SP.J.1001.2013.04257]

[88] Zhao J, Liu HW, Cui G, Yang XZ. Research on the software reliability model considering differences between testing profile and operational profile. Journal of Computer Research and Development, 2006,43(5):503−508 (in Chinese with English abstract). [doi: 10.1360/crad20060320]

[89] Zhao J, Liu HW, Cui G, Yang XZ. Software reliability model considering testing environment and actual operational environment. Journal of Computer Research and Development, 2006,43(5):881−887 (in Chinese with English abstract). [doi: 10.1360/crad20060517]

[90] Zhao J, Liu HW, Cui G, Yang XZ. Software reliability growth model with change-point and environmental function. Journal of Systems and Software, 2006,79(11):1578−1587. [doi: 10.1016/j.jss2006.02.030]

[91] Zhao J, Zhang RB, Gu GC. Study on software reliability growth model considering failure dependency. Chinese Journal of Computers, 2007,30(10):1713−1720 (in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2007.10.009]

[92] Xie JY, An JX, Zhu JH. NHPP software reliability growth model considering imperfect debugging. Ruan Jian Xue Bao/Journal of Software, 2010,21(5):942−949 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3539.htm [doi: 10.3724/SP.J.1001.2010.03539]

[93] Li HF, Li QY, Lu MY, A software reliability growth model considering an S-shaped testing effort function under imperfect debugging. Journal of Harbin Engineering University, 2011,32(11):1460−1467 (in Chinese with English abstract). [doi: 10.3969/j.issn.1006-7043.2011.11.012]

[94] Li HF, Li QY, Lu MY. Software reliability modeling with logistic test coverage function. Journal of Computer Research and Development, 2011,48(2):232−240 (in Chinese with English abstract).

[95] Pham H. System Software Reliability. London: Springer-Verlag, 2007.

[96] Pham H, Nordmann L, Zhang X. A general imperfect-software-debugging model with S-shaped fault-detection rate. IEEE Trans. on Reliability, 1999,48(2):169−175. [doi: 10.1109/24.784276]

[97] Pham H, Zhang X. NHPP software reliability and cost models with testing coverage. European Journal of Operational Research, 2003,145(2):443−454. [doi: 10.1016/S0377-2217(02)00181-9]

[98] Peng R, Hu QP, Ng SH, Xie M. Testing effort dependent software FDP and FCP models with consideration of imperfect debugging. In: Guerrero JE, ed. Proc. of the IEEE 4th Int'l Conf. on Secure Software Integration and Reliability Improvement (SSIRI). Singapore: IEEE, 2010. 141−146. [doi: 10.1109/SSIRI.2010.14]

[99] Hossain SA, Dahiya RC. Estimating the parameters of a non-homogeneous Poisson-process model for software reliability. IEEE Trans. on Reliability, 1993,42(4):604−612. [doi: 10.1109/24.273589]

[100] Huang CY, Lyu MR. Optimal testing resource allocation, and sensitivity analysis in software development. IEEE Trans. on Reliability, 2005,54(4):592−603. [doi: 10.1109/TR.2005.858099]

[101] Gokhale SS, Trivedi KS. Log-Logistic software reliability growth model. In: Sipple RS, ed. Proc. of the IEEE 3rd Int'l Symp. on High-Assurance Systems Engineering. Washington: IEEE, 1998. 34−41. [doi: 10.1109/HASE.1998.731593]

[102] Ohba M. Software reliability analysis models. IBM Journal of Research and Development, 1984,28(4):428−443. [doi: 10.1147/rd.284.0428]

[103] Wood A. Predicting software reliability. Computer, 1996,29(11):69−77. [doi: 10.1109/2.544240]

[104] Musa JD, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction and Application. New York: McGraw Hill, 1987.

[105] Pham H. Software Reliability. Singapore: Springer-Verlag, 2000. [doi: 10.1002/047134608X.W6952]

[106] Jha PC, Gupta D, Yang B, Kapur PK. Optimal testing resource allocation during module testing considering cost, testing effort and reliability. Computers & Industrial Engineering, 2009,57(3):1122−1130. [doi: 10.1016/j.cie.2009.05.001]

[107] Huang CY, Lin CT, Kuo SY, Lyu MR, Sue CC. Software reliability growth models incorporating fault dependency with various debugging time lags. In: Titsworth FM, ed. Proc. of the IEEE 28th Annual Int'l Conf. on Computer Software and Applications (COMPSAC). Hong Kong: IEEE, 2004. 186−191. [doi: 10.1109/CMPSAC.2004.1342826]

[108] Gokhale SS, Mullen RE. Queuing models for field defect resolution process. In: O'Conner L, ed. Proc. of the IEEE 17th Int'l Symp. on Software Reliability Engineering (ISSRE 2006). Raleigh: IEEE, 2006. 353−362. [doi: 10.1109/ISSRE.2006.38]

[109] Gokhale SS, Lyu MR, Trivedi KS. Reliability simulation of component-based software systems. In: Werner B, ed. Proc. of the IEEE 9th Int'l Symp. on Software Reliability Engineering. Paderborn: IEEE, 1998. 192−201. [doi: 10.1109/ISSRE.1998.730882]

[110] Lin CT, Huang CY, Sue CC. Measuring and assessing software reliability growth through simulation-based approaches. In: Ceballos S, ed. Proc. of the IEEE 31st Annual Int'l Conf. on Computer Software and Applications (COMPSAC). Beijing: IEEE, 2007. 439−448. [doi: 10.1109/COMPSAC.2007.141]

[111] Gokhale SS, Lyu MRT. A simulation approach to structure-based software reliability analysis. IEEE Trans. on Software Engineering, 2005,31(8):643−656. [doi: 10.1109/TSE.2005.86]

[112] Lyu MR. Handbook of Software Reliability Engineering. New York: McGraw Hill, 1996.

[113] Zhang C, Cui G, Bian YL, Liu HW. Component based software reliability process simulation considering imperfect debugging. High Technology Letters, 2014,20(1):9−15. [doi: 10.3772/j.issn.1006-6748.2014.01.002]

[114] Hou CY, Cui G, Liu HW, Zhou LK. A hybrid queueing model with imperfect debugging for component software reliability analysis. Intelligent Automation and Soft Computing, 2011,17(5):559−570. [doi: 10798587.2011.10643170]

[115] Pham H. An imperfect-debugging fault-detection dependent-parameter software. Int'l Journal of Automation and Computing, 2007, 4(4):325−328. [doi: 10.1007/s11633-007-0325-8]

[116] Zhang X, Teng X, Pham H. Considering fault removal efficiency in software reliability assessment. IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, 2003,33(1):114−120. [doi: 10.1109/TSMCA.2003.812597]

[117] Yamada S, Hishitani J, Osaki S. Software-Reliability growth with a Weibull test-effort: A model and application. IEEE Trans. on Reliability, 1993,42(1):100−106. [doi: 10.1109/24.210278]

[118] Ehrlich W, Prasanna B, Stampfel J, Wu J. Determining the cost of a stop-test decision. IEEE Software, 1993,10(2):33−42. [doi: 10.1109/52.199726]

[119] Zhang X, Pham H. Software field failure rate prediction before software deployment. Journal of Systems and Software, 2006,79(3): 291−300. [doi: 10.1016/j.jss.2005.05.015]

[120] Stringfellow C, Andrews AA. An empirical method for selecting software reliability growth models. Empirical Software Engineering, 2002,7(4):319−343. [doi: 10.1023/A:1020515105175]

[121] Zhang X, Pham H. A software cost model with warranty cost, error removal times and risk costs. IIE Transactions, 1998,30(12): 1135−1142. [doi: 10.1080/07408179808966570]

[122] Bai CG, Hu QP, Xie M, Ng SH. Software failure prediction based on a Markov Bayesian network model. Journal of Systems and Software, 2005,74(3):275−282. [doi: 10.1016/j.jss.2004.02.028]

[123] Cai KY, Cao P, Dong Z, Liu K. Mathematical modeling of software reliability testing with imperfect debugging. Computers & Mathematics with Applications, 2010,59(10):3245−3285. [doi: 10.1016/j.camwa.2010.03.011]

[124] Teng X, Pham H. A software cost model for quantifying the gain with considering of random field environments. IEEE Trans. on Computers, 2004,53(3):380−384. [doi: 10.1109/TC.2004.1261844]

[125] Shen CX, Zhang HG, Wang HM, Wang J, Zhao B, Yan F, Yu FJ, Zhang LQ, Xu MD. Research and development of trusted computing. Science China: Information Science, 2010,40(2):139−166 (in Chinese with English abstract).

[126] Zhang C, Cui G, Fu ZC. Overview of trustworthiness measurement mechanism and model in TCG. Journal of Harbin Institute of Technology, 2013,45(1):72−77 (in Chinese with English abstract).

[127] Amin A, Grunske L, Colman A. An approach to software reliability prediction based on time series modeling. Journal of Systems & Software, 2013,86(7):1923−1932. [doi: 10.1016/j.jss.2013.03.045]

[128] Kim T, Lee K, Baik J. An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm. Journal of Systems and Software, 2015,102:134−144. [doi: 10.1016/j.jss.2015.01.001]

[129] Tickoo A, Kapur PK, Khatri SK, Verma AK. Optimal release time determination for multi upgradation SRGM with faults of different severity using genetic algorithm. In: Proc. of the 4th Int'l Conf. on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions). Noida: IEEE, 2015. 1−6. [doi: 10.1109/ICRITO.2015.7359322]

[130] Jin C, Jin SW. Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization. Applied Soft Computing, 2016,40:283−291. [doi: 10.1016/j.asoc.2015.11.041]

[131] Sharma K, Garg R, Nagpal CK, Garg RK. Selection of optimal software reliability growth models using a distance based approach. IEEE Trans. on Reliability, 2010,59(2):266−276. [doi: 10.1109/TR.2010.2048657]

[132] Rana R, Staron M, Berger C, Hansson J, Nilsson M, Törner F, Meding W, Höglund C. Selecting software reliability growth models and improving their predictive accuracy using historical projects data. Journal of Systems and Software, 2014. [doi: 10.1016/j.jss.2014.08.033]

附中文参考文献:

[9] 张策,崔刚,刘宏伟,孟凡超,傅忠传.软件测试资源与成本管控和最优发布策略.哈尔滨工业大学学报,2014,46(5):51−58. [doi: 10.11918/j.issn.0367-6234.2014.05.009]

[10] 张策,崔刚,刘宏伟,孟凡超.构件软件可靠性过程技术.计算机学报,2014,37(12):2586−2612. http://www.cnki.net/kcms/detail/11.1826.TP.20140828.1915.003.html

[39] 王金勇,吴智博,舒燕君,张展.故障检测率不规则变化的软件可靠性模型.软件学报,2015(10):2465−2484. http://www.jos.org.cn/1000-9825/4746.htm [doi: 10.13328/j.cnki.jos.004746]

[73] 候春燕,陈晨,王劲松,林胜.考虑不完美排错情况的软件可靠性过程仿真.高技术通讯,2014,24(6):558−564. [doi: 10.3772/j.issn.1002-0470.2014.06.002]

[87] 李海峰,王栓奇,刘畅,郑军,李震.考虑测试工作量与覆盖率的软件可靠性模型.软件学报,2013,24(4):749−760. http://www.jos.org.cn/1000-9825/4257.htm [doi: 10.3724/SP.J.1001.2013.04257]

[88] 赵靖,刘宏伟,崔刚,杨孝宗.考虑测试与运行差别的软件可靠性增长模型研究.计算机研究与发展,2006,43(5):503−508. [doi: 10.1360/crad20060320]

[89] 赵靖,刘宏伟,崔刚,杨孝宗.考虑测试环境和实际运行环境的软件可靠性增长模型.2006,43(5):881−887. [doi: 10.1360/crad20060517]

[91] 赵靖,张汝波,顾国昌.考虑故障相关的软件可靠性增长模型研究.计算机学报,2007,30(10):1713–1720. [doi: 10.3321/j.issn:0254-4164.2007.10.009]

[92] 谢景燕,安金霞,朱纪洪.考虑不完美排错情况的 NHPP 类软件可靠性增长模型.软件学报,2010,21(5):942–949. http://www.jos.org.cn/1000-9825/3539.htm [doi: 10.3724/SP.J.1001.2010.03539]

[93] 李海峰,李秋英,陆民燕.考虑 S 型测试工作量函数与不完美排错的软件可靠性模型.哈尔滨工程大学学报,2011,32(11):1460–1467. [doi: 10.3969/j.issn.1006-7043.2011.11.012]

[94] 李海峰,李秋英,陆民燕.基于 Logistic 测试覆盖率函数的软件可靠性建模研究.计算机研究与发展,2011,48(2):232–240.

[125] 沈昌祥,张焕国,王怀民,王戟,赵波,严飞,余发江,张立强,徐明迪.可信计算的研究与发展.中国科学:信息科学,2010,40(2):139–166.

[126] 张策,崔刚,傅忠传.TCG 下可信度量机制与模型分析.哈尔滨工业大学学报,2013,45(1):72–77. [doi: 10.11918/j.issn.0367-6234.2013.01.014]

张策(1978－),男,吉林永吉人,博士,讲师,CCF 专业会员,主要研究领域为软件测试,软件可靠性评估,容错计算,可信计算.

孟凡超(1974－),男,博士,副教授,CCF 高级会员,主要研究领域为模型驱动的体系结构,软件重构与复用,企业资源计划.

考永贵(1971－),男,博士,教授,博士生导师,主要研究领域为随机微分方程理论与应用,变结构控制和容错控制,复杂网络与计算复杂性.

吕为工(1967－),男,副教授,主要研究领域为系统可靠性测评,容错计算.

刘宏伟(1971－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件可靠性分析,容错计算,移动计算.

万锟(1994－),男,主要研究领域为可靠性建模.

蒋家楠(1996－),女,主要研究领域为软件可靠性建模.

崔刚(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为高可靠计算,容错计算,嵌入式系统的研究及设计.

刘子和(1995－),女,主要研究领域为可靠性建模与评测.