软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006085]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

可靠性模型中故障检测率研究述评：建模、类型及效用分析*

张 策[1], 刘宏伟[2], 白 睿[1], 王瞰宇[1], 王金勇[3], 吕为工[1], 孟凡超[1]

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

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

[3](山西大学 软件学院,山西 太原 030006)

通讯作者: 张策, E-mail: zhangce@hitwh.edu.cn

摘 要: 故障检测率 FDR(Fault Detection Rate)是可靠性研究的关键要素,对于测试环境构建、故障检测效率提升、可靠性建模和可靠性增长具有重要作用,对于提高系统可靠性与确定发布时间具有重要现实意义.首先,对基于NHPP(Non-Homogeneous Poisson Process, 非齐次泊松过程)类的软件可靠性增长模型 SRGM(Software Reliability Growth Mode)进行概述,给出了建模本质、功用与流程.基于此,引出可靠性建模与研究中的关键参数——FDR,给出定义,对测试环境描述能力进行分析,展示不同模型的差异.着重剖析了 FDR 与失效强度、冒险率(风险率)的区别, 得出三者之间的关联性表述.全面梳理了 FDR 的大类模型,分别从测试覆盖函数视角、直接设定角度、测试工作量函数参与构成方式三个方面进行剖析,继而提出统一的 FDR 相关的可靠性模型.考虑到对真实测试环境描述能力需要,建立不完美排错框架模型,衍生出不完美排错下多个不同 FDR 参与的可靠性增长模型.进一步,在 12 个真实描述应用场景与公开发表的失效数据集上进行实验,验证不同 FDR 模型相关的可靠性模型效用,对差异性进行分析与讨论.结果表明,FDR 模型自身的性能可以支撑可靠性模型性能的提升.最后,指出了未来研究趋势和需要解决的问题.

关键词: 可靠性; 故障检测率; 测试覆盖率; 不完美排错; 效用分析

中图法分类号: TP311

中文引用格式: 张策,刘宏伟,白睿,王瞰宇,王金勇,吕为工,孟凡超.可靠性模型中故障检测率研究述评: 建模、类型及效用分析究.软件学报. http://www.jos.org.cn/1000-9825/6085.htm

英文引用格式: Zhang C, Liu HW, Bai R, Wang KY, Wang JY, Lv WG, Meng FC. Review on Fault Detection Rate in Reliability Model: Modeling, Type and Effectiveness Analysis. Ruan Jian Xue Bao/Journal of Software, 2020 (in Chinese). http://www.jos.org.cn/1000-9825/6085.htm

Review on Fault Detection Rate in Reliability Model: Modeling, Type and Effectiveness Analysis

ZHANG Ce[1], LIU Hong-Wei[2], BAI Rui[1], WANG Kan-Yu[1], WANG Jin-Yong[3], LV Wei-Gong[1], MENG Fan-Chao[1]

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

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

3(School of Software Engineering, Shanxi University, Taiyuan 030006, China)

Abstract: FDR(Fault Detection Rate),as the key element of reliability research, has great importance in constructing the test environment, improving fault detection efficiency, and modeling and improving reliability. In the meantime, it has important practical

* 基金项目: 国家科技支撑计划资助项目(2014BAF07B02); 国家自然科学基金(61473097);山东省重点研发计划项目(GG201703130116, GG201703040002); 威海市科技发展计划项目(ITEAZMZ001807)

Foundation item: National Science and Technology Support Program(2014BAF07B02); National Natural Science Foundation of China(61473097);Key R&D Program Project in Shandong Province(GG201703130116, GG201703040002); Weihai Science and Technology Development Plan Project(ITEAZMZ001807)

收稿时间: 2019-10-05; 修改时间: 2020-02-07, 2020-04-12; 采用时间: 2020-05-18; jos 在线出版时间: 2020-05-26

significance for improving system reliability and determining release time. First, the software reliability growth model SRGM (Software Reliability Growth Mode) based on NHPP (Non-Homogeneous Poisson Process) is summarized, and the essence, function and process of modeling are given. Second, based on this, FDR, the key parameter in reliability modeling and researching, is derived, and the definition of it is given. The test environment description ability is analyzed and differences of different models are shown. Third, emphasis is placed on the difference between FDR, failure strength and hazard rate (risk rate), and then the correlation between the three is derived. Next, the general model of FDR is comprehensively analyzed, which is analyzed from three perspectives of test coverage function, FDR set directly, and FDR constituted by testing effort function. Then a unified FDR-related reliability model is proposed. Considering the ability to describe the real test environment, the imperfect debugging framework model is established, and the reliability growth model of multiple different FDRs under imperfect debugging is derived. Further, experiments are carried out on 12 publicly available failure data sets describing real application scenarios to verify the effectiveness of reliability models related to different FDR models, and to analyze and discuss the differences. The results show that the performance of the FDR model can support the performance improvement of the reliability model. Finally, the trend of researches and the problems to be solved are pointed out.

Key words: reliability; fault detection rate; test coverage; imperfect debugging; effectiveness analysis

软件作为人工制品,是多类型软件开发人员协同完成的系统.由于软件是对物理系统和过程的计算机编程语言描述,因此其实际功能与预期成效很可能存在不一致之处.例如,软件自身隐藏的错误（Error）在特定情况下可能会导致故障（Fault）,而故障可能会导致失效（Failure）：软件的缺陷（Defect 可能是设计或编码时引入）在运行时会产生错误,当错误或故障积累到一定数量,或者达到某种条件时都会造成软件系统的失效.因此,软件的质量问题至关重要,尤其是可靠性问题,长久以来一直得到研究人员的关注.

可靠性作为软件的非功能质量属性,其可以通过软件可靠性增长测试这一重要途径来不断获得提高.软件测试过程中,基于故障不断被检测并修复进而使得软件可靠性持续获得增长的事实,为可靠性研究提供了有效的切入点.软件可靠性增长模型 SRGM(Software Reliability Growth Model)[1-2]从软件失效的角度进行可靠性的建模,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（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）之间的定量函数模型.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,建立能够准确地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.目前,SRGM 已成为度量、预测与管控成本支出下可靠性的重要技术[3-4],是管控可靠性与系统发布的常见工具.文献[5]即阐述了一种基于当前软件调试工作流的特征进行可靠性增长分析的方法 DWA-SRGM,其可指出影响产品评估的因素与瓶颈,从而支持流程改进决策.

而在整个可靠性的研究中,故障检测率 FDR(Fault Detection Rate)作为累积检测故障数量的主要影响因素,是建立可靠性增长模型的关键要素,因此,其是提高可靠性所需考察的重要问题.经过多年发展,FDR 以及其支持的可靠性研究取得了重要进展.目前,国内外尚没有对 FDR 进行全面述评的综合性分析文章,本文在作者前期大量工作[1-2, 6-7]的基础上,基于国内外研究情况进行悉心梳理,从问题起源与功用、相关区别与联系、综合分类讨论、不完美排错下模型性能差异性分析等视角对 FDR 进行了全面述评,并进行了大量的实验验证,进而给出后续研究趋势和需要解决的问题,以期为研究人员提供有价值的参考与借鉴,促进可靠性研究取得新进展.

本文的贡献着重体现在对如下四个问题进行了深入回答：

（1）对 FDR 进行了全面深入的研究性论述,对 FDR 在可靠性研究中的功能、地位、作用和成效进行了深刻阐述,这在当前可靠性研究中尚属首次；

（2）厘清了 FDR 与失效强度、风险率/冒险率的区别与联系,从数学角度提出并证明了 FDR 与测试覆盖函数关系的重要定理；

（3）拓展了 FDR 的研究内涵,提出了典型的不完美排错环境下 FDR 相关的软件可靠性框架模型；

（4）通过在大量的真实应用场景上的综合实验,深入分析了 FDR 效用以及对可靠性模型的影响,为研究和选择 FDR 提供了有重要借鉴意义的参考.

本文第 1 节对以 SRGM 为核心的可靠性进行了概要介绍,引出故障检测率 FDR；第 2 节从可靠性模型构建视角给出了 FDR 的定义,从多个角度阐释其功用；第 3 节重点剖析了 FDR 与失效强度、风险率/冒险率的区

别与联系；第 4 节给出了分类视角下的 FDR 构成；第 5 节重点对 FDR 性能及其对 SRGM 的影响进行分析,提出不完美排错框架模型,通过数值分析研究模型差异性；最后指出了后续研究的趋势与需要解决的问题,给出结论.

1 软件可靠性增长模型建模——本质、功用与流程

软件开发过程中具有大量复杂的随机性与不确定性,随着研究人员对测试过程的不断深入认识,SRGM 的研究持续至今.SRGM 具备描述测试过程中失效发生、故障检测和修复等动态特征,其刻画了软件的一种故障行为.由于 SRGM 建立了故障失效个数与可靠性间的数学关联模型,因此,利用 SRGM 就可以计算特定时刻与可靠性紧密相关的参数指标,包括失效个数、失效率、失效间隔,以及可靠性等；进而,可以对测试资源进行动态的调配,预测当前测试环境下软件达到预期目标（例如,可靠性）时所需要的时间（被称为发布时间）、成本（被称为发布成本）等重要信息.

在研究内容上,从突破早期完美排错的限制,到仅考虑到新故障引入的研究,以及考虑测试工作量或测试覆盖率的 NHPP 类软件可靠性建模框架的研究,再到涵盖不完全排错与新故障引入等各类不完美排错的研究,进一步拓展至变动点问题、测试资源分配问题、最优发布问题等,越来越多的可靠性模型得以建立.在求解方法与技术上,从建立单一微分方程(组)的简单或复杂的解析式方法,扩展到非解析式方法、排队论技术和最优化方法,进而采用离散事件仿真与非参量求解方法等,这些正在推动软件可靠性研究不断走向深入.图 1 对 SRGM 的建模与功用进行了展示.

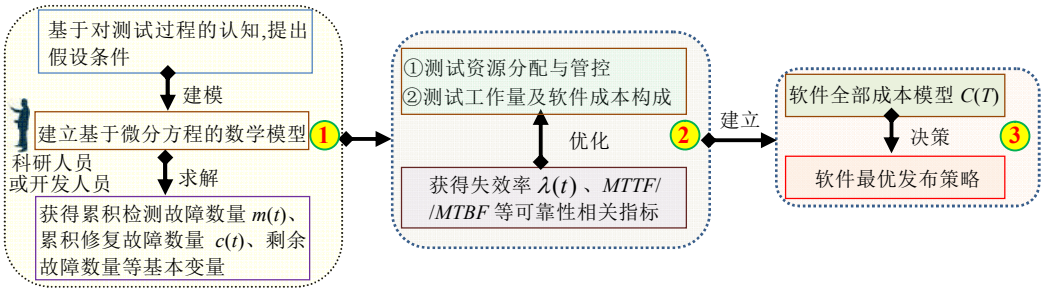


Fig.1 Modeling essence and function of SRGM

图 1 软件可靠性增长模型的建模本质与功用

若 SRGM 将软件测试过程视为若干个随机过程的统一,失效发生后,测试工作量 TE 被定量地消耗用以进行故障的检测、隔离、排除等.随着测试的不断进行,软件中的故障不断减少,从而软件可靠性得到不断的增长.软件可靠性能够通过软件可靠性增长模型 SRGM 进行有效度量与预测.目前,SRGM 得到了快速的发展,现已被广泛应用,成为定量评估软件可靠性的数学工具.图 2 对 SRGM 的建模流程进行了归纳.

在代表 NHPP(Non-homogeneous Poisson process, 即非齐次泊松过程)类 SRGM 研究起源的经典 G-O 模型[8]中,Goel L 与 Okumoto K 建立了类似于下面的微分方程:

$$\frac{dm(t)}{dt}=\lambda(t)=b(t)\big(a(t)-m(t)\big) \tag{1}$$

该微分方程具有更一般性,其假设认为,t 时刻累积检测的故障数量 $dm(t)/dt$ 与此刻软件中剩余的故障数量 $(a(t)\text{-}m(t))$成正比例,比例系数 $b(t)$为该时刻的故障检测率 FDR(Fault Detection Rate).显然,当 $b(t)=b$, $a(t)=a$ 时,(1)式转化为 G-O 模型,该模型虽忽视了故障修复与新故障引入情况,但却成为日后研究连续性 NHPP 类 SRGM 中共同被遵守的事实.这其中,FDR 的表现形式是 $b(t)$,从宏观上对故障检测能力进行描述,但并未指出支撑故障检测能力的构成.并且,G-O 模型认为 FDR 为常量（在软件测试的过程中保持不变）,这显然与真实的测试过程差异较大.此外,文献[9]建立了一个非齐次泊松过程类软件可靠性增长模型——Bbell-SRGM；文献[10-11]提出了结合软件运行环境中每单位时间故障检测率的不确定性的软件可靠性模型；文献[12]提出了具有时间相关性噪音影响下的 NHPP 类 SRGM；文献[13]提出了一种基于 Weibull 分布引进故障的 SRGM,使软件可靠性增长

模型更加符合实际的软件故障检测过程,具有重要的理论意义和实际价值.同时,文献[14]利用基于深度循环神经网络（RNN）编码器-解码器的深度学习模型来预测软件中的故障数量并评估软件的可靠性；文献[15]推导了一种可以在软件开发和运行的各个阶段提供良好的软件可靠性预测的模型,提供了更好的预测性能.文献[10-11,16]还提出了用于比较 SRGM 拟合优良性的诸多标准,例如均方误差 MSE、预测率风险 PRR、预测能力 PP 以及赤池信息量准则 AIC,为 SRGM 拟合度评估提供了有效方法.

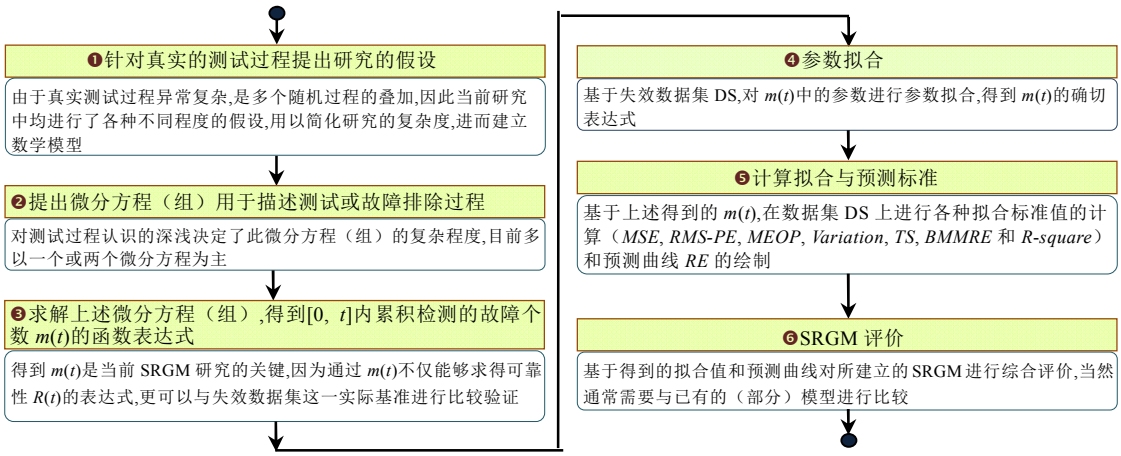


Fig.2 Modeling process of SRGM

图 2 SRGM 建模流程

特别指出,本文研究遵从 SRGM 研究领域中的默认规定：

（1）“error(错误)”[17]、“defect 缺陷”[18]、“fault(故障)”[19-20],以及“failure（失效）”[21]并没有被严格区分,表示的含义是等价的,这与容错研究中对“defect→error→fault→failure”进行严格区分是不同的.这是因为,SRGM 研究中隐含着由缺陷、错误到失效的一一对应关系,因而将其等同看待.

（2）这里并不区分修正、改正、修复、排除,四者均表示故障被排除掉.

2 故障检测率 FDR——SRGM 建模的关键要素

2.1 SRGM中关键参数

在当前所建立的众多 SRGM 中,均包含“在 t 时刻这一当前时间点上,累积检测到的故障数量 $m(t)$与当前软件中剩余的故障数量成正比例”这一基本假设,此比例系数被称为故障检测率 FDR（Fault Detection Rate,习惯用 $b(t)$来表示）.因此,软件中全部初始故障数量 $a(t=0)=a$ 和故障检测率函数 FDRF(Fault Detection Rate Function,通常用 $b(t)$加以表示)是影响 SRGM 的重要因素[20],这已成为 SRGM 研究中的共识.图 3 展示了经典的完美排错 G-O 模型的建模过程,其中的 FDRF 为常量 b.

在可查证的 SRGM 研究文献[17-20.22-29]中,FDR 是 SRGM 建模中必不可少的参数,其描述了故障被检测出来的能力,同时也是对测试效能的一种描述.

定义 故障检测率 FDR：即故障检测率函数,表示当前时刻单位时间内单个故障被检测到的平均概率[9],或每个故障的查出率[30],通常用 $b(t)$来表示.

FDR 表征了测试环境下故障被查出的效率,具有描述综合测试策略的能力,因而与包括测试人员、测试技术、测试工具等构成的整体测试环境紧密相关.

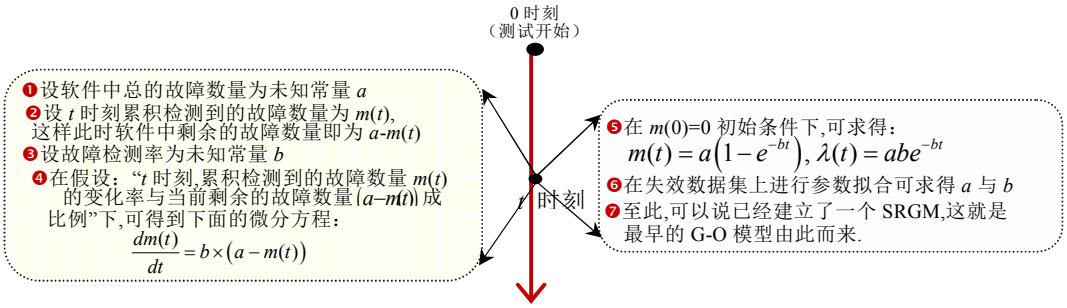


Fig.3 SRGM modeling analysis——taking the classic G-O model as an example

图 3 SRGM 建模解析——以经典的 G-O 模型为例

从早期提出将 FDR 看作常数的软件可靠性增长模型,到提出整体呈现递减趋势的幂函数类型 FDR,再到能够基本刻画测试环境平缓变化的 *S* 型 FDR,以及(复杂)指数类型 FDR 的研究,整体上对 FDR 的研究方向呈现出贴近工程实际化特点,因而能够更好地描述测试环境的改变,帮助提高可靠性模型的性能.

2.2 测试环境描述能力

测试过程的目的是不断发现与修复故障,提高可靠性,达到预期（发布）要求.客观上,测试环境的不同,以及测试人员实施测试策略的差异,将使得不同系统工程在测试中表现出不同的外在特征.从建立数学模型的角度来看,式(1)描述了测试过程中关于故障检测的共性,但不同模型的区别与 FDR: $b(t)$关联紧密.可见,FDR 能够从整体上刻画测试效果,这使其成为影响 SRGM 性能的主要评测点.

关于对 FDR 影响因素的研究,Huang C Y 在他的一系列文献[31-32]中指出,测试初始阶段,很多故障可以被检测出来,测试 FDR 依赖于故障发现效率、故障密度、测试工作量、检出率这些参数；中期阶段,FDR 通常还依赖于上述因素以外的其他因素,包括 CPU 指令的执行率、失效与故障间关联、代码膨胀系数、测试团队技能、程序规模、软件可测试性,以及每日历天预定的 CPU 执行小时等因素.因此,当需求发生改变和新的特征被添加,或者修复期间有新故障引入时,FDR 可能会发生改变.此外,考虑到故障检测的随机性和复杂性,即针对其是具有较强不确定性的概率事件这一方面,文献[9]提出了一个响铃形的故障检测率函数,文献[33]考虑了受操作环境影响的系统故障检测率,文献[34]提出了一种随时间不规则变化的故障检测率,均体现了其突变性.

2.2.1 从 FDR 角度研究 CP

在关于移动点/拐点 CP(Change-Point)的研究上,也均将 FDR 作为分段研究实施的对象.由于 FDR 刻画了整个测试环境下的综合测试效果,因而当测试环境发生变化与转折时,FDR 就会有所变动.这样,将 FDR 进行数学上的 CP 处理[35-38]（即 CP 前后 FDR 函数形式发生变化）成为考虑 CP 的 SRGM 研究惯例.

由于 FDR 与测试环境下的失效分布情况等多因素有关联,而失效分布又会受到例如运行环境、测试策略和资源分配等[32]多种因素的影响.因此,当测试策略和测试资源分配发生变化时,CP 就会出现.此外,增加对程序的认识和自动化测试工具的运用也会引发 CP.因此,FDR 在整个测试过程中会发生改变.

2.2.2 从 FDR 角度研究测试环境与运行环境差异

此外,在考虑测试与运行环境的差异研究上[27,39],差异点也是从 FDR 角度来进行研究的,相关的研究还涉及故障减少因子[20,40-43]、学习效果[17]、测试压缩因子[44]、测试覆盖率[45-46]等.事实上,CP 具有描述测试环境与运行环境差异性的能力.

2.3 FDR基本类型

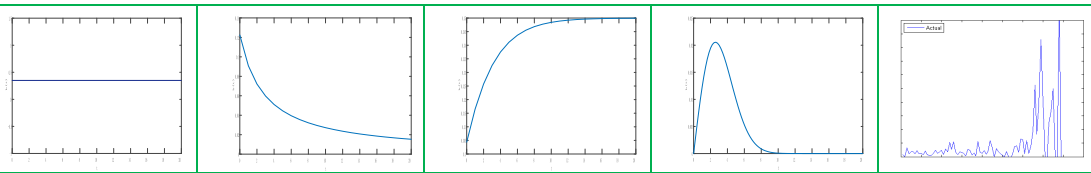
FDR 的构成上,存在着 $b(t)=b$, $b(t)=b^2t/(1+bt)$[26,47], $b(t)=\dfrac{b}{1+\beta e^{-bt}}$ [28,48], $b(t)=b\alpha\beta e^{-\beta t}$ [49], $b(t)=b\alpha\beta t e^{-\beta t^2/2}$ [49]等多种函数形式,基本情况如表 1 所示.

Table 1 Fault Detection Rate Function $b(t)$

表 1 故障检测率函数 $b(t)$

类型	故障检测率 $b(t)$	概要分析
常量类型	$b(t)=b$	常量,认为 FDR 在整个测试过程中并不发生改变,这虽能带来求解上的简易,但显然偏离实际测试情形,无法描述 $b(t)$随测试环境的变化情况,在早期研究中被经常使用
幂函数类型	$b(t)=b^2t/(1+bt)$	整体呈现递减的趋势
S 型	$b(t)=b/(1+\beta e^{-bt})$	呈现 S 型变化趋势,能够基本刻画测试环境的简单变化,即能够描述测试中 FDR 对多种测试环境的适应
	$b(t)=\frac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)t}}$	
(复杂)指数类型	$b(t)=b\alpha\beta e^{-\beta t}$	整体呈现递减的趋势
	$b(t)=b\alpha\beta t e^{-\beta t^2/2}$	呈现先增后减的整体趋势,且带有波峰,波峰后急剧下降趋近于零

图 4 给出了表 1 中五类 $b(t)$的基本形状,其中设定 b=0.35, $\alpha=0.25$, $\beta=0.15$, $t\in[0, 20]$,仅用以展示曲线形状.



(a)常函数 (b)减函数 (c)增函数 (d)先增后降 (e)故障检测率随测试时间不规则变化的情况

Fig.4 Types of FDR: constant function; subtraction function; increasing function; increasing first, then decreasing function; irregularly changing function (unstable change)

图 4 FDR 类型：常函数、减函数、增函数、先增后降、不规则变化(不平稳变化)

3 FDR 与失效强度、风险率/冒险率的区别与联系

3.1 FDR

FDR 是与时间相关的故障检测率函数,用以表示单位时间内故障被发现的概率,为了深入研究,$b(t)$有时被研究人员设定为 t 的某种函数.

3.2 失效强度

失效强度通常用 $\lambda(t)$ 来进行,表示在 t 时刻单位时间内失效发生的次数,在 SRGM 的研究中用以表示累积故障检测数量的导数,即 $\lambda(t)=\frac{dm(t)}{dt}$.可见, $\lambda(t)$ 是跳跃强度函数,即在时刻 t 单位时间内失效发生的次数.如果故障出现导致失效发生而立即被发现的话,$b(t)=z(t)$,否则二者并不相等.

3.3 风险率/冒险率

风险率即风险函数(hazard function),通常用 $h(t)$表示,即程序正确地运行到时刻 t 时发生故障的概率.$h(t)$是每个故障失效发生率或冒险率[50],即“瞬时失效率”.事实上,$h(t)$作为“瞬时失效率”是一种冒险率.失效率的定义是：

$$[t, t+\Delta t]\text{内失效率}=\frac{\Pr\{\text{在区间}[t, t+\Delta t]\text{内发生失效}\mid\text{在}t\text{不发生失效}\}}{(t+\Delta t)-t=\Delta t} \tag{2}$$

对上式求 Δt 的极限,求得的就是冒险率,即“瞬时失效率”,可见冒险率或“瞬时失效率”是失效率的一种时间上的逼近.

“设 T 表示从 0 开始运行一个程序,到程序发生失效为止经历的时间”,T 是失效时间的独立随机变量,其失效分布函数和失效密度函数分别为 $F(t)$和 $f(t)$,则可得如下两式：

$$\begin{aligned} h(t)&=\lim_{\Delta t\to 0}\frac{\Pr\{t+\Delta t\ge T>t\mid T>t\}}{\Delta t}=\lim_{\Delta t\to 0}\frac{\Pr\{t+\Delta t\ge T>t\}}{\Pr(T>t)\Delta t}=\lim_{\Delta t\to 0}\frac{F(t+\Delta t)-F(t)}{[1-F(t)]\Delta t}\\ &=\frac{dF(t)}{dt}\times\frac{1}{[1-F(t)]}=\frac{f(t)}{[1-F(t)]}=\frac{f(t)}{R(t)}\end{aligned} \tag{3}$$

$$h(t)=\frac{f(t)}{1-F(t)}=\frac{f(t)}{R(t)} \tag{4}$$

则可以得到 $R(t+\Delta t)$ 的表达式、$R(t)$的导数与 $h(t)$和 $R(t)$的关系如下：

$$\begin{aligned} &R(t+\Delta t)=\Pr\{T>(t+\Delta t)\}=\Pr\{(T>t)\wedge\text{在区间}[t, t+\Delta t]\text{内程序不发生故障}\}\\ &=\Pr\{(T>t)\}\times\Pr\{\text{在区间}[t, t+\Delta t]\text{内程序不发生故障}\}=R(t)[1-h(t)\Delta t]\end{aligned} \tag{5}$$

$$R(t+\Delta t)=R(t)[1-h(t)\Delta t]\Rightarrow\frac{R(t+\Delta t)-R(t)}{\Delta t}=-R(t)h(t)\Rightarrow\frac{dR(t)}{dt}=-h(t)R(t) \tag{6}$$

因此,求解上式,可得： $R(t)=e^{\left[-\int_0^t h(x)dx\right]}$, 显然, $R(0)=1$; $R(\infty)=0$.

3.4 区别与联系

综上,我们可以得出,故障检测率 FDR、失效强度以及风险率/冒险率均为可靠性相关指标,三者均与程序运行环境中的多因素（例如,程序规模、故障密度、故障测试效率、测试工作量、CPU 指令的执行率、代码膨胀系数等）相关联.故障检测率 $b(t)$与失效强度 $\lambda(t)$ 和风险率/冒险率 $h(t)$均成正比例关系.具体讨论如下：

（1）故障检测率 $b(t)$描述了测试人员在测试策略的指导下采用合适的测试案例和测试方法检测出故障的能力.其与 t 时刻点上的测试覆盖率 $c'(t)$成正比例,是测试覆盖 $c(t)$的函数,可表示为 $b(t)=c'(t)/(1-c(t))$,与可靠性 $R(t)$直接相关.

（2）失效强度可用于描述 t 时刻累积检测的故障数量 $m'(t)$,其与此刻软件中剩余的故障数量$(a(t)-m(t))$成正比例,比例系数 $b(t)$为该时刻的故障检测率 FDR.则根据 G-O 模型的基本假设,可得失效强度 $\lambda(t)$ 与故障检测率 $b(t)$的关系如下式：

$$\lambda(t)=\frac{dm(t)}{dt}=b(t)[a-m(t)] \tag{7}$$

（3）风险率/冒险率描述了程序正确地运行到时刻 t 时发生故障的概率,其侧重于每个故障失效发生率或冒险率[37],即“瞬时失效率”,为失效率的一种时间上的逼近.在 SRGM 研究中,根据上述式（7）的表示以及假设条件可知：风险函数 $h(t)$与失效强度 $\lambda(t)$ 相等,即 $h(t)=\lambda(t)$.从而根据 G-O 模型的基本假设,可进一步得到风险函数 $h(t)$与故障检测率 $b(t)$的关系如下式：

$$b(t)=\frac{\lambda(t)}{(a-m(t))}=\frac{h(t)}{(a-m(t))} \tag{8}$$

同时,文献[52]认为测试覆盖下的故障以常量概率 k 被检测出来,从而可推导出：

$$\lambda(t)=\left[\frac{c'(t)}{1-c(t)}\right](a-m(t)) \tag{9}$$

则 $h(t)$为：

$$h(t)=\frac{c'(t)}{1-c(t)} \tag{10}$$

此时,即可得到风险函数或每个故障的失效发生率 $h(t)$与故障检测率 $b(t)$相等,即 $b(t)=h(t)$.

4 分类与综合视角下的 FDR

4.1 测试覆盖函数视角———FDR与测试覆盖函数的关系

软件测试过程中,随着测试用例的不断执行,测试覆盖的范围不断扩大,故障被检测出的可能性不断提高.测试覆盖范围的扩大,使得被检测出的故障数量增多,因此,测试覆盖函数与故障检测率关系密切.测试覆盖TC(Testing Coverage)指测试系统覆盖被测试系统的程度,描述了测试与被测试对象之间的相关性[51],测试覆盖率较高的程序在测试过程中执行了更多的源代码.这里,从代码语句覆盖的角度来看,可以定量表示为式(11)所示:

$$TC=\frac{S_c}{S_t} \tag{11}$$

其中 S_c 是测试案例覆盖的 KLOC(Kiloline of Code), S_t 是全部 KLOCs.

当考虑时间因素时,TC 就演变为随测试时间变化的测试覆盖函数 TCF(Testing Coverage Function): $c(t)$,可用以描述测试覆盖率的变动情况.显然,$c(t)$是一个非负的且非降的函数.

4.1.1 测试覆盖函数起源——采用设定 FDR 与测试覆盖率成比例的间接视角

针对上述 (11) 式的来源,文献[52]最早给出了严谨的分析,其认为测试覆盖下的故障以常量概率 k 被检测出来,从而建立了下面的微分等式:

$$\frac{dm(t)}{dt}=a_0\cdot c_d(t)\cdot\frac{dc(t)}{dt} \tag{12}$$

其中 a_0 表示初始时软件中的故障总数,$c_d(t)$表示 t 时刻测试覆盖下故障被检测出的概率,按照假设,得到 $c_d(t)=k$ 常量.这样,令 $a_0k=a$,则得到 $m(t)=ac(t)$,进而基于 $\lambda(t)=\frac{dm(t)}{dt}$ 的事实,可推导出:

$$\lambda(t)=\left[\frac{c'(t)}{1-c(t)}\right]\big(a-m(t)\big) \tag{13}$$

则冒险函数(The Hazard Function)或每个故障的失效发生率(The failure occurrence rate per fault)$h(t)$为:

$$h(t)=\frac{c'(t)}{1-c(t)} \tag{14}$$

这样,在通常情况下,可认为 $b(t)=h(t)$.通过上述得到的 $m(t)=ac(t)$,可以看出,这里就直接建立了 $m(t)$与 $c(t)$的关联.为此,将现有各种 $m(t)$函数表达式改写成 $m(t)=ac(t)$形式,即可获得相应的 $c(t)$函数.

显然,对于完美的测试覆盖,$c(t)$是 t 的增函数,且满足 $c(0)=0$, $c(\infty)=1$.

4.1.2 测试覆盖函数起源——采用与 SRGM 和 TEF 建模相同的直接视角

$c(t)$的建立是基于下面的假设:任意 t 时刻的测试覆盖率与当前剩余的测试覆盖值成比例.这样,可以得到下面的微分方程:

$$\frac{dc(t)}{dt}=\beta(t)\big(\alpha-c(t)\big) \tag{15}$$

其中 α 是可获得的测试覆盖目标值,考虑到测试后期软件中未覆盖代码的难以测试性,其满足 $0<\alpha<1$; $\beta(t)$为比例系数,也被称为测试覆盖成熟率,满足 $\beta(t)>0$. $\beta(t)$ 与测试人员的技能紧密相关,因而假定测试案例设计人员的测试技能随着测试覆盖率的增长而增加.这样, $\beta(t)$ 被定义为:

$$\beta(t)=b_{\text{sta}}\left[r+(1-r)\left(\frac{c(t)}{\alpha}\right)\right] \tag{16}$$

其中 $r=b_{ini}/b_{sta}$,b_{ini} 和 b_{sta} 分别是初始和稳态时测试案例设计人员的测试技能因子.(16)式的初始条件为 $c(t)=0$,这样可求得:

$$c(t)=\frac{\alpha\left(1-e^{-b_{sta}t}\right)}{1+\beta e^{-b_{sta}t}} \tag{17}$$

其中, $\beta=(1-r)/r$.这样,当 r=1 时,$c(t)$呈现指数增长趋势；当 r=0 时,$c(t)$呈现 S 型增长趋势；其它情况下,$c(t)$受制于多种因素,变化趋势较为复杂.可以看出,这种建模方法和获得测试覆盖函数的思路与对 SRGM 和 TEF 的建模是完全一致的,在我们前期的研究[1,7]中已分别对 SRGM 和 TEF[6]的建模进行了综合研究,这里不再赘述.相应地,另一种融入测试覆盖的 NHPP 可靠性增长模型被建立如下：

$$\frac{dm(t)}{dt}\cdot\frac{1}{c'(t)}=b\left(a-m(t)\right) \tag{18}$$

其中,b 是每个故障与每个可获得测试覆盖下的故障检测率,在 $m(0)$=0 的初始条件下,可求得：

$$m(t)=a\left(1-e^{-bc(t)}\right) \tag{19}$$

易见,当 $c(t)$=1 时,模型演变为经典的 G-O 模型,这也意味着,G-O 模型假定测试覆盖满足 100%.

4.1.3 基于测试覆盖函数的 SRGM 建模

$c(t)$表示截止至 t 时刻已被测试的代码或案例所占的百分比（原表述不清晰）; 1-$c(t)$表示到 t 时刻尚未被测试的代码所占的比例.显然,$c(t)$的导数 $c'(t)$表示 t 时刻时的测试覆盖率.易知,FDR 与 $c'(t)$成正比例,且与 1-$c(t)$成反比例.这样,文献[11, 40]认为 $c'(t)/(1-c(t))$可被用来度量 t 时刻的故障检测率 $b(t)$.

$$b(t)=\frac{c'(t)}{1-c(t)} \tag{20}$$

因为基于 $c(t)$的 SRGM 研究,通常以下面的公共假设[19,53-54]为基础,从而依据这些假设可建立如(21)式所示的微分方程,其可用以描述测试过程中故障检测与测试覆盖的基本关系：

$$\frac{dm(t)}{dt}=\left(\frac{c'(t)}{1-c(t)}\right)\cdot[a(t)-m(t)] \tag{21}$$

相比于基于故障检测率进行 SRGM 研究的事实,易知很多研究中对 $b(t)$的函数形式进行直接设定会引发更多的随机性.上式 $b(t)$从测试覆盖函数 TCF：$c(t)$进行定义,由于测试覆盖可借助测试用例执行经过的结构或数据流单元[55]来度量,这使得 FDR 的函数形式依赖于从测试覆盖角度进行确定的结果,如(22)式所示：

$$m(t)=e^{-B(t)}\left[m_0+\int_{t_0}^{t}a(\tau)e^{B(\tau)}\left(\frac{c'(\tau)}{1-c(\tau)}\right)d\tau\right] \tag{22}$$

其中

$$B(t)=\int_{t_0}^{t}\left(\frac{c'(\tau)}{1-c(\tau)}\right)d\tau=\int_{t_0}^{t}b(\tau)d\tau \tag{23}$$

4.1.4 测试覆盖函数与故障检测率函数和可靠性的关联

在提出 Logistic TEF[56-57]用以描述 TE 消耗的基础上,有先增后减变化趋势的 S 型函数又被用以刻画测试覆盖率函数[58-60],并与 SRGM 结合来度量可靠性.由于 TC 表征了被测试代码的比例,易知该比例的升高会使得更多的故障被检测出来,进而随着故障的修复,可靠性将得到提高.可见,$c(t)$与 t 时刻的可靠性 $R(t)$具有一定的定量关系,而探讨二者之间的关联在可靠性研究中[53-55,61-63]已持续十余年.

可以看出,故障检测率 $b(t)$直接描述了测试环境下故障被检测出的一种概率,其是测试人员在测试策略的指导下,采用合适的测试案例或测试方法对故障进行检测的能力描述；测试覆盖 $c(t)$注重于测试过程中测试代码对被测试软件/源代码的覆盖程度.这样,易知故障检测率是测试覆盖的函数,即 $b(t)=f(c(t))$,例如 $b(t)=c'(t)/(1-c(t))$.表 2 即给出了典型的测试覆盖函数与故障检测率函数,并对二者的关联进行了初步分析.

Table 2 Typical test coverage function and fault detection rate function

表 2 典型的测试覆盖函数与故障检测率函数

对象	含义	与可靠性 $R(t)$关联	函数特征	典型函数	关联
测试覆盖函数：$c(t)$	描述测试用例覆盖被测试代码的比例	直接相关,例如：$R(h\mid t)=e^{-\int_t^{t+h}\lambda(s)ds}=e^{-a(c(t+h)-c(t))}$	非负非降的增函数,介于[0, 1]之间	$\left(1-e^{-bt}\right)$ [52] $\left(1-e^{-btc}\right)$ [8, 52] $\left(1-(1+bt)e^{-bt}\right)$ [47,52]	$b(t)=c'(t)/(1-c(t))$

Table 2 Typical test coverage function and fault detection rate function(Continued)

表 2 典型的测试覆盖函数与故障检测率函数(续)

对象	含义	与可靠性 $R(t)$关联	函数特征	典型函数	关联
故障检测率函数：$b(t)$	描述故障被检测出的概率	直接相关	非负非降，介于[0, 1]之间	$b(t)=b^2t/(1+bt)$[26,47] $b(t)=b/(1+\beta e^{-bt})$[28] $b(t)=b\alpha\beta te^{-\beta t^2/2}$ [49]	$b(t)=c'(t)/(1-c(t))$

表 2 注：表 2 中无论 $c(t)$还是 $b(t)$均是已有研究工作中直接提出来的函数形式,进而在真实的测试数据集上进行验证,尚没有从严格理论分析的角度进行证明,属于后验式.

显然,随着测试的进行,$c(t)$不断增长,但由于 $b(t)$与 $c(t)$之间关系较为复杂,$b(t)$的变化形式难以直接预测.从定量关系上来看,$b(t)$与 $c(t)$都在[0, 1]之间,这里以 $c(t)=\dfrac{\alpha\left(1-e^{-b_{sta}t}\right)}{1+\beta e^{-b_{sta}t}}$ 为例,提出如下定理：

定理：当 β 与 α 的大小关系确定时,$b(t)$与 $c(t)$存在明确的大小关系.

证明：$b(t)=c'(t)/(1-c(t))$, $c(t)=c(t)(1-c(t))/(1-c(t))$, 令 $z(t)=b(t)-c(t)=[c'(t)-c(t)-c^2(t)]/(1-c(t))$.对 $c(t)$求导数可得,$c'(t)=\dfrac{\alpha be^{-bt}\left(1+\beta\right)}{\left(1+\beta e^{-bt}\right)^2}$,代入 $z(t)=b(t)-c(t)=[c'(t)-c(t)-c^2(t)]$,得：$z(t)=\dfrac{\alpha be^{-bt}\left(1+\beta\right)}{\left(1+\beta e^{-bt}\right)^2}-\dfrac{\alpha\left(1-e^{-bt}\right)}{1+\beta e^{-bt}}-\dfrac{\alpha^2\left(1-e^{-bt}\right)^2}{\left(1+\beta e^{-bt}\right)^2}$,经化简得：$z(t)=\dfrac{\alpha\left(\beta-\alpha\right)\left(e^{-bt}\right)^2+\alpha\left(b+b\beta+1-\beta+2\alpha\right)e^{-bt}-\alpha\left(\alpha+1\right)}{\left(1+\beta e^{-bt}\right)^2}$.令 $z(t)=0$, $e^{-bt}=x$,因上式分母恒大于零,且 $\alpha\neq 0$,上式可转化为：$\left(\beta-\alpha\right)x^2+\left(b+b\beta-\beta+1+2\alpha\right)x-\left(1+\alpha\right)=0$,根据判别式公式可得：$\Delta=\left[b\left(1+\beta\right)+\left(1-\beta\right)+2\alpha\right]^2+4\left(\beta-\alpha\right)\left(1+\alpha\right)$,化简得：$\Delta=\left(b^2+1\right)\left(1+\beta\right)^2+2b\left(1+\beta\right)\left(1-\beta+2\alpha\right)$,因为 $0<\beta<1$,所以 $\Delta>0$,所以上式有两个根.根据一元二次方程求根公式得：$x_1=\dfrac{-\left(b+b\beta+1-b+2\alpha\right)-\sqrt{\Delta}}{2\left(\beta-\alpha\right)}$, $x_2=\dfrac{-\left(b+b\beta+1-b+2\alpha\right)+\sqrt{\Delta}}{2\left(\beta-\alpha\right)}$.易见两个根的分子恒小于零,下面对 β 与 α 的关系进行讨论：

当 $\beta>\alpha$ 时,$x_1<x_2<0$,函数图像开口向上,此时图像的大致曲线如图 5 所示.此时可得：

(1)$x_1<x<x_2$ 时,即 $\ln\left(\dfrac{1}{x_2}\right)^{\frac{1}{b}}<t<\ln\left(\dfrac{1}{x_1}\right)^{\frac{1}{b}}$ 时 $z(t)<0$；

(2)$x<x_1$ 或 $x>x_2$ 时,即 $t>\ln\left(\dfrac{1}{x_1}\right)^{\frac{1}{b}}$ 或 $t<\ln\left(\dfrac{1}{x_2}\right)^{\frac{1}{b}}$ 时,$z(t)>0$.

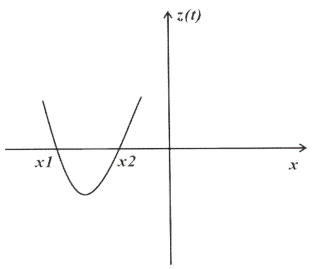


Fig.5 Curve shape with opening up

图 5 开口向上曲线形状

当 $\beta<\alpha$ 时,$0<x_1<x_2$,函数图像开口向下,此时图像的大致曲线如图 6 所示.此时可得：

(1)$x_1<x<x_2$ 时,即 $\ln\left(\frac{1}{x_2}\right)^{\frac{1}{b}}<t<\ln\left(\frac{1}{x_1}\right)^{\frac{1}{b}}$ 时,$z(t)>0$；

(2)$x<x_1$ 或 $x>x_2$ 时,即 $t>\ln\left(\frac{1}{x_1}\right)^{\frac{1}{b}}$ 或 $t<\ln\left(\frac{1}{x_2}\right)^{\frac{1}{b}}$ 时,$z(t)<0$.

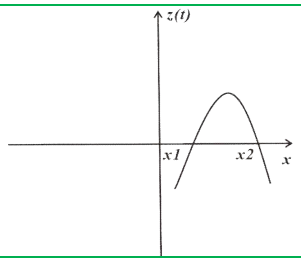


Fig.6 Curve shape with opening down
图 6 开口向下曲线形状

同理,当 $c(t)$为其他形式时可作同样的分析处理,进而得到有价值的结果.

4.2 直接设定FDR

由于 FDR 构成的不同,$b(t)$存在着多种函数形式.早期研究中认为 FDR 为常量,设定 $b(t)=b$,显然无法描述 $b(t)$随测试环境的变化情况.随着研究的深入,FDR 已呈现出多种函数形式,例如 $b(t)=b^2t/(1+bt)$, $b(t)=\frac{b}{1+\beta e^{-bt}}$, $b(t)=b\alpha\beta e^{-\beta t}$, $b(t)=b\alpha\beta te^{-\beta t^2/2}$ 等.这些 FDR 可分为两类：

①常量类型：$b(t)=b$,认为 FDR 在整个测试过程中并不发生改变,这虽能带来求解上的简易,但显然偏离实际测试情形；

②时变类型：例如 $b(t)=\frac{c}{1+\alpha e^{-bt}}$,其呈现 S 型变化趋势,能够描述测试中 FDR 对多种测试环境的适应.

另一方面,鉴于 FDR 是对测试环境的直接描述,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.文献[64]即基于各种 FDR,将其引入到 SRGM 建模中进行研究.

4.3 复合式——TE参与的FDR

Li QY 在她的文献[65]中指出,考虑 TE 的可靠性建模可以进一步改善 SRGM 的拟合和预测效果.测试过程中,随着故障检测与修复等环节中 TE 的不断消耗,软件可靠性不断得到提高.在我们前期研究[6-7]的基础上,对 TEF 进行了梳理,并给出了考虑 TEF 的 SRGM 研究.例如,在如下的假设下：“[0, t]内累计检测到的故障数量与当前剩余的故障数量下所花费的测试工作量 TE 时的故障检测率 $b(t)$成比例”,可以得到典型的考虑 TE 的 SRGM 建模方法[31-32,36,64,66-73]：

$$\frac{dm(t)}{dt}\cdot\frac{1}{w(t)}=b(t)\cdot[a-m(t)] \tag{24}$$

其中,$b(t)$为故障检测率函数,$m(t)$表示[0, t]内累计检测到的故障数量,a 表示软件中的总故障个数.在本质上,(24)式的故障检测率函数为 $b(t)\cdot w(t)$.实际上,FDR 表示“单位 TE 花费下平均检测出的故障”[64],这样 $b(t)\cdot w(t)$实际上是一个复合函数,其包含了测试工作量的因素.

4.4 小结

在前述分析的基础上,我们可以将多种描述故障检测能力的函数称为故障检测因子 $k(t)$,$k(t)$存在如下四种情况：

（1）$k(t)=b(t)=b$——常量；

（2）$k(t)=b(t)$——此时 $b(t)$存在多种函数形式,如前所述；

（3）$k(t)=\frac{c'(t)}{1-c(t)}$ ——从测试覆盖的角度,提出了故障被检测出来的能力；

（4）$k(t)=b(t)\cdot\omega(t)$ ——从测试工作量与故障检测率的复合角度,也即在当前测试资源消耗的情况下,提出了故障被检测出来的能力；

（5） $k(t)=b\cdot c'(t)$ ——从测试覆盖与每个故障可获得测试覆盖下的故障检测率角度.

综上,针对故障检测的描述,式(23)建立了统一的故障检测模型——t 时刻检测的故障与当前软件中剩余的故障数量成比例,同时该模型也指出了 t 时刻累积修复的故障数量函数形式,即 $c(t)=p(t)\cdot m(t)$.

$c(t)$描述了测试用例的执行情况,侧重于测试覆盖代码的程度；$w(t)$描述了测试资源的消耗情况,侧重于测试成本的花销；$b(t)$描述了测试策略的效果,侧重于测试人员的技能、工具与技术.此三者从不同角度对测试环境效果进行了建模描述,在当前软件可靠性建模中用以描述故障被检测出来的能力.

无论是故障检测率 FDR: $b(t)$,还是故障检测因子 $k(t)$,都是对测试过程中故障被检测出的程度或效果的描述,刻画了在测试环境下消耗测试资源以执行测试策略来对故障进行检测的能力.因此,其与测试环境下的多因素（失效分布情况、故障密度、程序大小、测试人员技能、测试案例与工具等）有关联.

5 FDR 性能及其对可靠性模型影响分析

迄今为止,国内外尚未对 FDR 影响下的可靠性模型进行研究,也缺少对 FDR 自身性能的分析,本节将通过在大量真实失效数据集上进行实验验证来分析这两个方面.

5.1 统一的故障检测框架模型

在当前可查证到的 SRGM 研究中,所有模型的建立均是基于下面的假设,即“t 时刻累积检测的故障数量与当前软件中剩余的故障数量成比例”,这个比例即是故障检测因子.为此,我们提出统一的故障检测过程建模：

$$\frac{dm(t)}{dt}=k(t)\left[a(t)-p(t)\cdot m(t)\right] \tag{25}$$

这里认为 t 时刻检测的故障数量与当前剩余的故障总数成比例,比例系数是故障检测因子 $k(t)$（实际上,这里的 $k(t)$已经超过了 SRGM 研究中单纯的故障检测率 $b(t)$,其表示当前测试环境下测试人员在消耗测试资源,执行测试案例进行测试时故障被检测出来的概率（函数）,是涵盖多个测试因素的综合性指标,但从保持延续性以及便于实验等因素考虑,这里依旧命名为 $b(t)$,后面不再专门用 $k(t)$进行表示）.$p(t)$表示故障修复概率,表示 t 时刻被检测出的故障被修复的比率.

在 $m(0)=0$ 和 $a(0)=a$ 的初始条件下,求解可得到：

$$m(t)=e^{-\int_0^t b(x)\cdot p(x)dx}\int_0^t e^{\int_0^y b(x)\cdot p(x)dx}a(y)b(y)dy \tag{26}$$

显然,式(26)中 $b(t)$, $p(t)$和 $a(t)$的多种设置,可以得到多种 $m(t)$,因此,这里提出的是一种框架式模型,具有较强的柔韧性.

文献[17,19,27-28]中提出了较为常用的 FDR,以此式(26)为基础,通过设定不同形式的 $b(t)$可以得到在相应测试环境下的可靠性模型（即累积故障检测数量）,具体求解情况如下：

（1）若 $b(t)=\dfrac{b}{1+\beta e^{-bt}}$,则可以求得 $m(t)$如下：

$$m(t)=e^{\int_0^t -\frac{be^{bx}p(x)}{\beta+e^{bx}}dx}\int_0^t \frac{be^{-\int_0^y -\frac{be^{bx}p(x)}{\beta+e^{bx}}dx}e^{by}a(y)}{b+e^{by}}dy \tag{27}$$

此时,随着测试的持续进行, $b(t\to\infty)\to b$.

（2）若 $b(t)=b^2t/(1+bt)$,则可以求得 $m(t)$如下：

$$m(t)=e^{\int_0^t -\frac{b^2xp(x)}{bx+1}dx}\int_0^t \frac{b^2ye^{-\int_0^y -\frac{b^2xp(x)}{bx+1}dx}a(y)}{by+1}dy \tag{28}$$

（3）若 $b(t)=b\alpha\beta e^{-\beta t}$,则可求得 $m(t)$如下：

$$m(t)=e^{\int_0^t -\alpha b\beta e^{-\beta x}p(x)dx}\int_0^t \alpha b\beta e^{-\int_0^y -\alpha b\beta e^{-\beta x}p(x)dx}e^{-\beta y}a(y)dy \tag{29}$$

此时,随着测试的持续进行, $b(t\to\infty)=0$.

（4）若 $b(t)=b\alpha\beta te^{-\beta t^2/2}$,则可以求得 $m(t)$如下：

$$m(t)=e^{\int_0^t -\alpha b\beta x e^{-\frac{\beta x^2}{2}}p(x)dx}\int_0^t \alpha b\beta y e^{-\int_0^y -\alpha b\beta x e^{-\frac{\beta x^2}{2}}p(x)dx}e^{-\frac{\beta y^2}{2}}a(y)dy \tag{30}$$

此时,随着测试的持续进行, $b(t\to\infty)=0$.

（5）若 $b(t)=\dfrac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)t}}$,则可以求得 $m(t)$如下：

$$m(t)=e^{-b\int_0^t \frac{e^{bx(\sigma+1)}p(x)}{\sigma+e^{bx(\sigma+1)}}dx(\sigma+1)}\int_0^t \frac{b\int_0^y \frac{e^{bx(\sigma+1)}p(x)}{\sigma+e^{bx(\sigma+1)}}dx(\sigma+1)}{\sigma+e^{by+b\sigma y}}\left(be^{by+b\sigma y}a(y)+b\sigma e^{by+b\sigma y}a(y)\right)dy \tag{31}$$

此时,随着测试的持续进行, $b(t\to\infty)\to b(1+\sigma)$.

5.2 模型与验证所用的失效数据集

虽然已有众多 SRGMs 被提出,但多以指数型和 *S* 型模型[47,74]为主,同时为了观测不同 FDR 的差异,表 3 列出了用于参与比较的可靠性模型和本文衍生的模型(不失一般性,这里所提出的框架模型中设定 $p(t)=p$, $a(t)=a$)及 FDR.

Table 3 Reliability models and FDRs involved in comparison

表 3 参与比较的可靠性模型及 FDR

模型	类型	累积故障检测数量 $m(t)$	FDR 类型
M-0[8]	完美排错+常量型 FDR	$m(t)=a\left[1-e^{-bt}\right]$, $b(t)=b$, 经典的 G-O 模型	常量,认为 FDR 在整个测试过程中并不发生改变
M-1	不完美排错+常量型 FDR	$m(t)=\dfrac{a-ae^{-bpt}}{p}$, $b(t)=b$	
M-2	不完美排错+指数型 FDR	$m(t)=\dfrac{a-ae^{\alpha bpe^{\beta t}}e^{-\alpha bp}}{p}$, $b(t)=b\alpha\beta e^{-\beta t}$.	递减的指数变化趋势
M-3[49]	完美排错+指数型 FDR	$m(t)=a\left[1-e^{-b\alpha\left(1-e^{-\beta t}\right)}\right]$, $b(t)=b\alpha\beta e^{-\beta t}$.	
M-4	不完美排错+弯曲 *S* 型 FDR	$m(t)=\dfrac{a-ae^{p\ln(\beta+1)}e^{-p\ln\left(\beta+e^{bt}\right)}}{p}$, $b(t)=\dfrac{b}{1+\beta e^{-bt}}$.	弯曲 *S* 型函数, 且非降
M-5[49]	完美排错+弯曲 *S* 型 FDR	$m(t)=\dfrac{a\left(1-e^{-bt}\right)}{1+\beta e^{-bt}}$; $b(t)=\dfrac{b}{1+\beta e^{-bt}}$.	
M-6	不完美排错+先增后减型 FDR	$m(t)=\dfrac{a-ae^{\alpha bpe^{-\frac{\beta t^2}{2}}}e^{-\alpha bp}}{p}$, $b(t)=b\alpha\beta te^{-\beta t^2/2}$.	先增后减的变化趋势,波峰后急剧下降
M-7[75]	完美排错+先增后减型 FDR	$m(t)=a\left[1-e^{-b\alpha\left(1-e^{-\beta t^2/2}\right)}\right]$, $b(t)=b\alpha\beta te^{-\beta t^2/2}$.	
M-8	不完美排错+递减型 FDR	$m(t)=\dfrac{a-ae^{p\left(\ln(bt+1)-bt\right)}}{p}$, $b(t)=b^2t/(1+bt)$.	呈现递减的趋势
M-9[47]	完美排错+递减型 FDR	$m(t)=a\left[1-(1+bt)e^{-bt}\right]$, $b(t)=b^2t/(1+bt)$.	
M-10	不完美排错+复杂弯曲 *S* 型 FDR	$m(t)=\dfrac{a-ae^{-p\ln\left(\sigma+e^{bt+b\sigma t}\right)}e^{p\ln(\sigma+1)}}{p}$, $b(t)=\dfrac{b(1+\sigma)}{1+\sigma e^{-b(1+\sigma)t}}$.	复杂弯曲 *S* 型函数, 且非降, σ 为测试过程学习因子, 可正可负

表 3 注：在已有的文献中,FDR 多以 $b(t)$[11, 19]进行表示,此外也存在用 $r(t)$和 $d(t)$[9]进行表示的情况.这里将此 3 种形式统一为更为常见的 $b(t)$式样.

这 11 个模型,可以分为 6 组,其中前 5 组均是在同一 FDR 下采用不同的建模假设所得到的模型,可以用于比较不同模型对 FDR 的影响,最后一组仅包含复杂弯曲 *S* 型函数 M-10；同时,M-1, M-2, M-4, M-8, M-10 是统

一建模框架下不同 FDR 衍生出的模型,可以观测不同 FDR 对模型的影响.

为了验证与比较模型的性能,我们遴选了 12 个失效数据集 DS_1—DS_{12}[76-79,18,24,80-81,48,19,82-83]开展实验工作.这些失效数据集由失效检测时间 *ti*(通常是以周为单位)和累积检测的失效故障数量 *yi* 构成,其均来自国际上著名的计算机公司公开发布的计算机(软件)系统在测试过程中搜集的真实数据,描述了不同的测试场景,从而作为可靠性模型验证的载体,得到了广泛的认可与应用.表 4 列出了来自于真实应用场景下的 12 个失效数据集,对其构成与来源等进行介绍.

Table 4 Failure data set in real application scenario

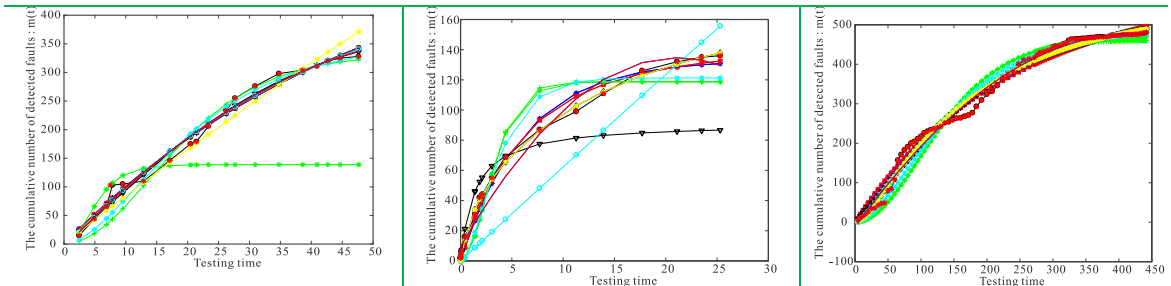
表 4 真实应用场景下的失效数据集

数据集	真实来源案例(公司或单位软件/程序/系统/项目)	测试对象规模(公司或单位软件/程序/系统/项目)	失效或故障记录时间	累计检测到的失效或故障个数(记录开始至记录结束)
DS_1[76]	IBM:数据库应用软件	1317,000(代码行数)	记录 19 周	15~328
DS_2[77]	RADC(罗马航空发展中心)的 T1 系统:实时命令与控制应用	21,700(代码行数)	记录 21 周:由贝尔实验室的 9 名工程师负责收集	2~136
DS_3[78]	在线数据采集软件包	40,000(代码行数)	记录 21 天	2~46
DS_4[79]	AT&T Bell 实验室:网络管理系统	不明	记录 680.02 个 CPU 单位时间	1~22
DS_5[18]	Tandem(美国天腾计算机)公司:计算机工程项目	不明	记录 20 周	16~100
DS_6[24]	大型医疗记录系统	含 188 个软件构件	记录 18 周	28~176
DS_7[80]	Misra 系统	不明	记录 25 小时	27~136
DS_8[81]	Misra 系统	1.5 百万行代码级别当量	记录 38 周	15~231
DS_9[48]	海军舰队计算机程序中心:海军战术数据系统	含 38 个工程模块	记录 849 天	1~34
DS_{10}[19]	Tandem Computer(美国天腾计算机)公司	几百万行代码级别当量	记录 19 周	1~42
DS_{11}[82]	航空程序	9564 行 C 语言代码	记录 21 周	2~403
DS_{12}[83]	电信系统	不明	1 个标准化时间	0.05~1(标准化处理之后故障个数)

5.3 性能验证与分析

5.3.1 拟合性能分析

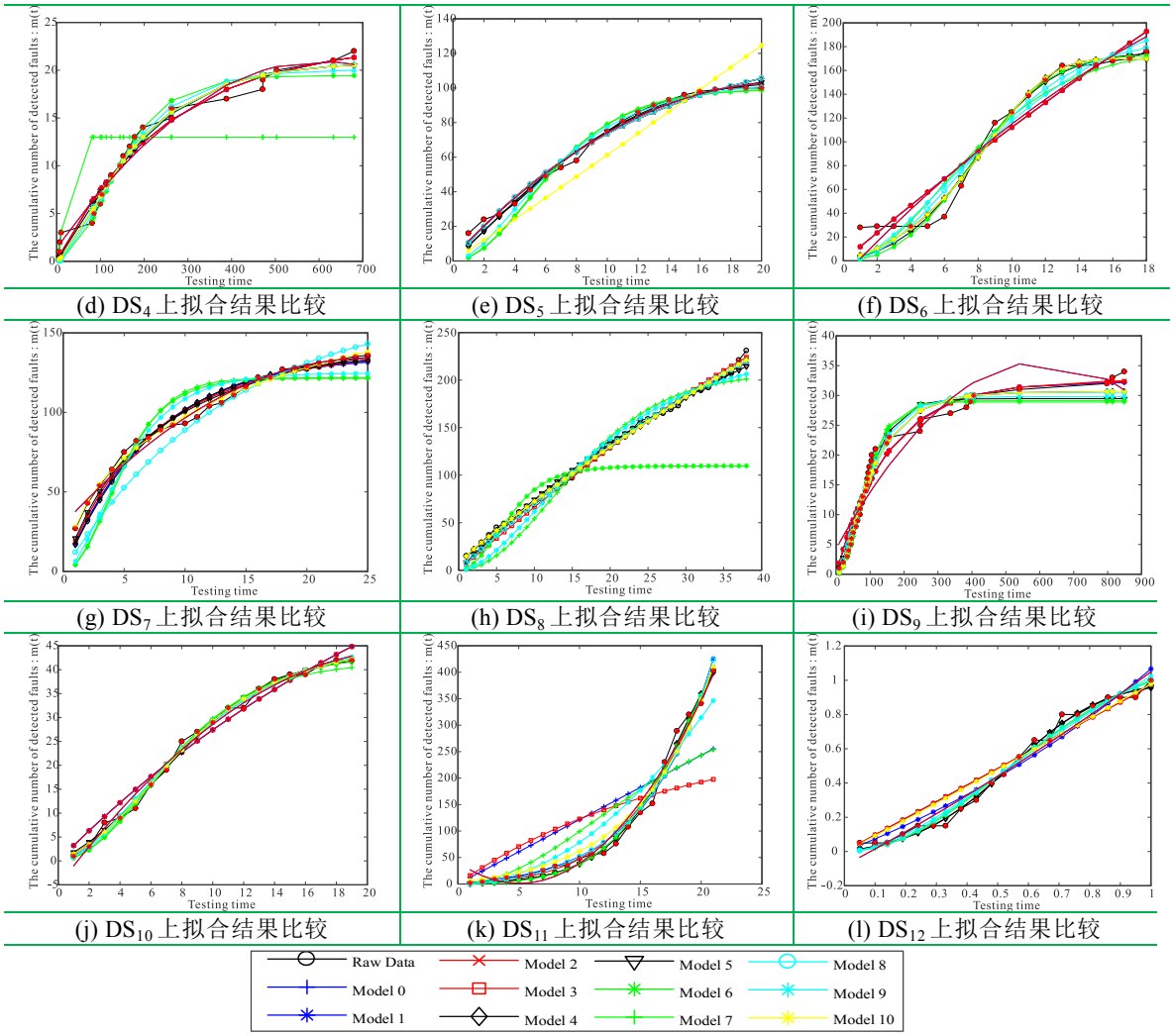
为了获得更为广泛的实验结果以得到有价值的分析,这里在 12 个公开发表的失效数据集上进行实验验证(更多实验结果可联系作者).基于模型在 12 个数据集上的拟合结果,我们绘制了参与比较的模型与真实的失效数据数值之间的拟合曲线,如图 7 所示.



(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

Fig.7 Fitting curve of M-0 to M-10 on DS1~DS12

图 7　M-0 至 M-10 在 DS1~DS12 上度量拟合曲线



(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

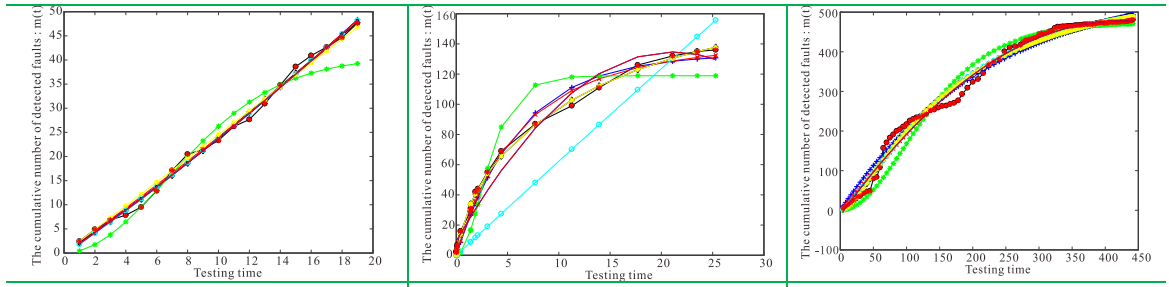
(g) DS_7 上拟合结果比较　(h) DS_8 上拟合结果比较　(i) DS_9 上拟合结果比较

(j) DS_{10} 上拟合结果比较　(k) DS_{11} 上拟合结果比较　(l) DS_{12} 上拟合结果比较

Fig.7　Fitting curve of M-0 to M-10 on DS_1~DS_{12} (Continued)

图 7　M-0 至 M-10 在 DS_1~DS_{12} 上度量拟合曲线(续)

为更加清晰地观测不同 FDR 对模型带来的影响,在图 7 的基础上,我们特将不同 FDR 融入统一框架模型而衍生出的 M-1、M-2、M-4、M-6、M-8、M-10 绘制在一处,用以比较框架模型内不同子模型之间的差异,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

Fig.8　Fitting curves of M-1, M-2, M-4, M-6, M-8, M-10 on DS1~DS12

图 8 M-1、M-2、M-4、M-6、M-8、M-10 在 DS1~DS12 上度量拟合曲线

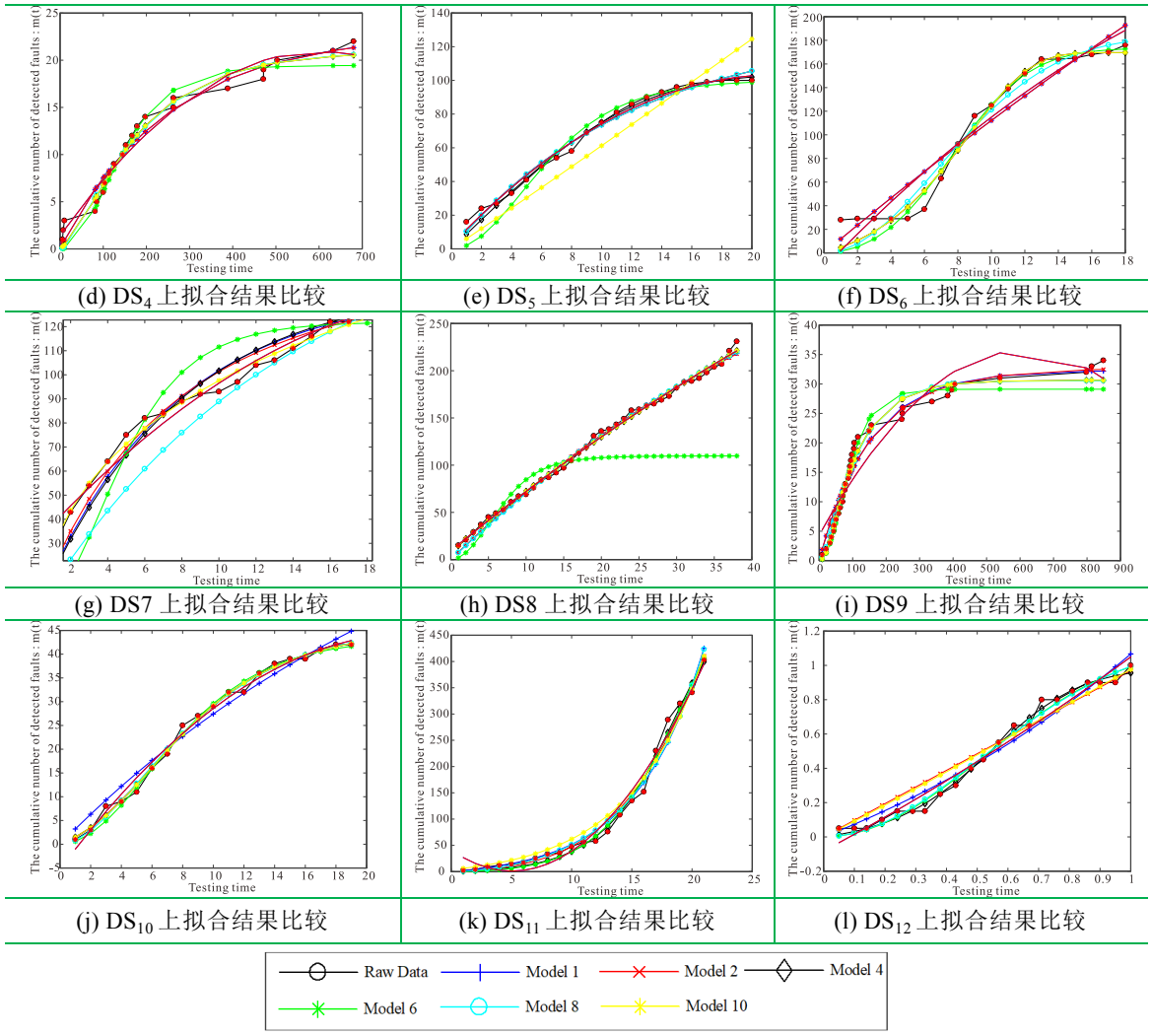


Fig.8 Fitting curves of M-1, M-2, M-4, M-6, M-8, M-10 on DS_1~DS_{12}(Continued)

图 8 M-1、M-2、M-4、M-6、M-8、M-10 在 DS_1~DS_{12} 上度量拟合曲线(续)

从图 7 以及图 8 展示的大量实验所呈现的系列曲线结果可以看出：

（1）整体上,除了部分模型出现严重偏差以外(例如,M-6 在 DS_1 上,M-8 在 DS_2 上,M-7 在 DS_4 上,M-6 在 DS_8 上等),大部分模型与真实的失效数据集的增长形状保持一致,这说明软件测试过程从累计故障检测的角度具有凹或凸指数型增长趋势,这也证明采用 NHPP 指数类失效时间模型研究软件可靠性增长具有现实合理性.

可以发现,弯曲 *S* 型 FDR 函数（包括复杂弯曲 *S* 型函数）对数据集的适应性最好,这通过 M-4、M-5 和 M-10 在众多失效数据集上的性能得到验证,这也与我们前期研究[1, 7]的结果保持一致,这是因为呈现 *S* 型的 FDR 函数能够适应不同测试阶段测试环境的变化,具有较强的柔韧性；M-6 和 M-7 所包含的先增后减型 FDR 也展现出了较好的性能,但并不具备 *S* 型 FDR 的强劲适应性；递减型与常数型 FDR 参与的模型性能表现一般,这主要是由于真实测试环境的变化并非稳定或连续变化.这些不同的实验现象可以解释为故障检测过程取决于特定测试策略下的测试技术、人员技能等实际因素,这使得 FDR 既不是常量,也不是持续递减,而是具有 *S* 型等变化规律,特别是对于大型复杂软件的长期测试过程来说更加满足这一变化规律.

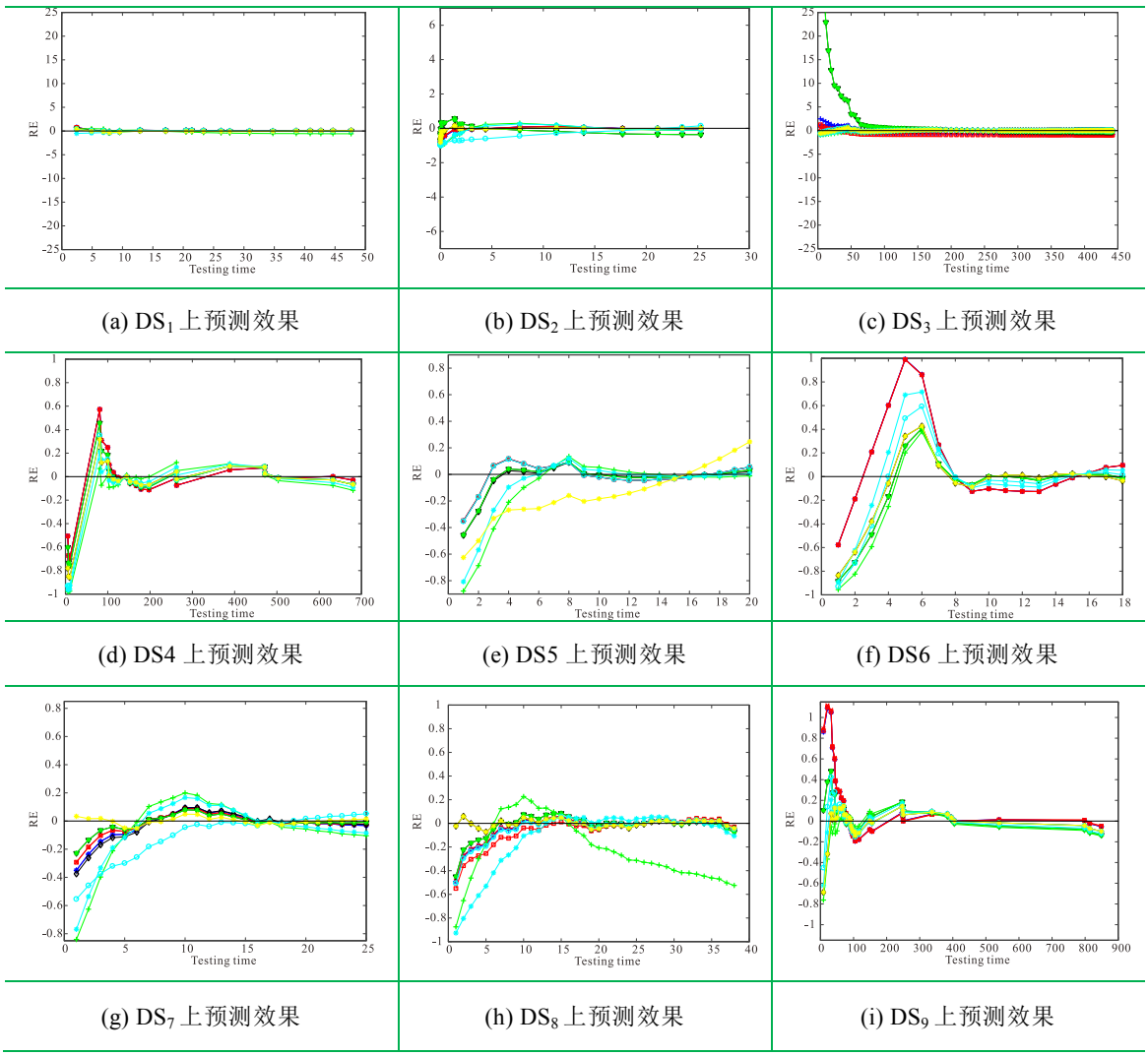
综上,从具有相同 FDR 的不同模型曲线进行分析可以看出,本文所提出的模型优于其他同组模型.

（2）虽然 *S* 型 FDR 性能较好,但将其融入某些建模假设较为合理的模型中,这些模型在有些失效数据集上不一定会表现出良好的性能.由软件自身的特点与测试过程的特点所决定,不同公司发布的失效数据集差异性较大,这使得很难存在某个模型在所有失效数据集上均表现出良好性能.例如,M-5 在 DS_2 上的性能并不理想,这与 Sharma K[22]等人提出的模型具有局限性相一致.

（3）另外,不完美排错模型考虑到了更多的实际情况（正如文献[84]所述,不完美排错模型将故障排除效率以及软件故障总数均看作随时间变化的函数,更具有实际应用意义）,因此其整体性能好于完美排错模型,因为真实的测试与故障检测和排错过程是被多种因素影响的复杂随机过程,具有不完美特性,因此,考虑到实际不完美情况的模型能够给建模带来更多的精准性.

5.3.2 预测性能分析

为了观测模型的预测性能,我们绘制了参与比较模型的预测 *RE* 曲线,如图 9 所示.*RE* 曲线越趋近于 0,表明预测性能越好,位于 0 以上是正向预测,位于 0 以下是负向预测.



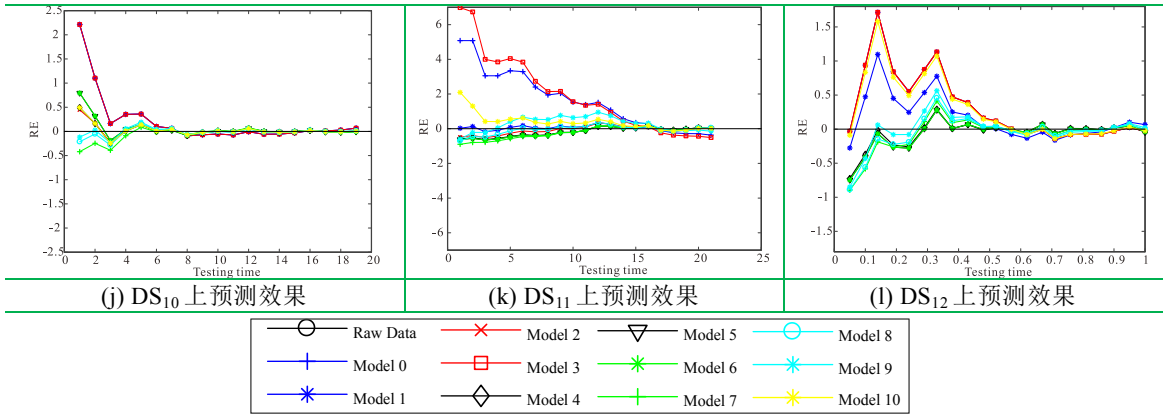
(a) DS_1 上预测效果　(b) DS_2 上预测效果　(c) DS_3 上预测效果

(d) DS4 上预测效果　(e) DS5 上预测效果　(f) DS6 上预测效果

(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　Prediction curve of M-0 to M-10 on DS_1~DS_{12}

图 9　M-0 至 M-10 在 DS_1~DS_{12} 上的预测曲线



(j) DS_{10} 上预测效果　(k) DS_{11} 上预测效果　(l) DS_{12} 上预测效果

Fig.9　Prediction curve of M-0 to M-10 on DS_1~DS_{12} (Continued)

图 9　M-0 至 M-10 在 DS_1~DS_{12} 上的预测曲线(续)

基于数据集的预测可以看作是模型对未来测试性能的描述能力,也反映出模型在后续时刻累积检测出故障的能力.从图 9 所展示的曲线走势以及相应的数据分析可以看出:

(1) 整体上,除了个别模型在部分数据集上出现预测偏差以外(例如,M-10 模型在 DS_5 上和 M-7 模型在 DS_8 上发生预测失真现象),大部分模型的预测曲线随着测试的进行都逐渐趋向于 0,表明效果较为理想;

(2) 模型本身的建模合理性对预测效果存在着较大影响,但预测依然会受到模型参数个数、数据集本身的数量大小等因素的影响,例如在测试前半程,预测曲线的剧烈起伏变化表明模型正在进行对数据的拟合适应;

(3) 具有(弯曲)*S* 型 FDR 函数的模型能够较快地趋于 0 水平线,表明其预测能力较好,这包括 M-4、M-5.同时,结合上述针对于图 7、图 8 的讨论可得,具有优秀适应能力的 *S* 型 FDR 能够帮助模型呈现更为强大的拟合与预测能力;

(4) M-10 在 DS_5 上的预测性能并不理想,这与其在 DS_5 上的拟合曲线所显示的拟合性能不理想相一致.从而可以得出,任何一个模型不能在所有的数据集上表现良好(包括拟合性能和预测性能),这种现象是数据集本身或模型参数数量过多等因素造成的;

(5) 呈现 *S* 型变化趋势的 $b(t)=\dfrac{b}{1+\beta e^{-bt}}$,在多个数据集的多个模型上保持良好的性能,表明实际测试环境存在移动点/拐点 CP,而并非平滑进行,因此包含 CP 的模型能够具有更好的性能.

5.4 讨论

基于前述 11 个模型在 12 个真实失效数据集上的大量实验结果分析,特做如下讨论:

(1) 整体上,可靠性模型不能适应于所有的数据集,且根据实验结果可观察出其拟合度量与预测具有相对的一致性,这两点与 FDR 的关联并不大,从本文大量的实验结果中没有得到明显关联支撑;

①可靠性模型的有效性具有较大的局限性,模型在一部分数据集上表现优秀的同时,在另外一些数据集上效果一般甚至较差,适用性受到较大限制;

②此外,模型的拟合性能与预测性能具有基本统一性.

(2) FDR 对模型的影响,特别是同一框架中不同的 FDR 对性能的影响存在较大差异:

①框架模型具有较强的柔韧性,如本文提出的一样,不同的 FDR 使得框架模型衍生为具体的模型,这为区分 FDR 的性能差异带来可能;

②同一 FDR 在不同可靠性模型中的性能通过模型的整体性能来反应,由于不同模型是研究人员基于不同的假设建模得到,因而难以进行有效衡量.

(3) FDR 建模具有客观性与主观性两个方面:

由于 FDR 与整体测试过程紧密相关,而测试过程是测试人员按照测试策略进行测试工作不断发现软件故

障的过程,是多个随机因素叠加在一起的随机过程,具有很强的随机性.因此,测试过程的随机性会直接为 FDR 的建模带来困难.

①对于具有较强测试规划,呈现一定规律的测试过程,FDR 往往具有明显的变化规律,这为FDR 建模提供依据;

②测试过程的复杂性解释了本文给出的当前研究中存在多种类型 FDR 形式的主要原因,具有符合多种实际的真实性.

（4）包含更多测试信息的发布将为 FDR 的建模与研究带来直接帮助,也为深入研究可靠性的增长与变动提供有效支持.

特别指出,FDR 受到多种因素影响,具有典型的随机性,例如不同测试策略、测试工具与方法等都会对其带来扰动影响.因此,现有研究中提出了多种不同形式的 FDR,且相互之间的差异性较大,但在特定的测试环境下还是具有合理性的.在实际测试过程中,由于测试都是在预定或已知的条件下主动实施的,此时完全可以根据测试整体安排来选择和确定 FDR.

6 研究挑战与趋势分析

6.1 面临的挑战

6.1.1 描述软件测试过程长期变化规律的 FDR 函数

FDR 是 SRGM 中最为重要的参变量,其描述了单位时间内被检测到的故障数量的变化情况,因而其在本质上刻画了整个测试过程中测试效率的演变,对于可靠性模型的演变尤其是增长至关重要.

从本文前述介绍中可以看出,FDR 实际上涉及到 SRGM 中的故障检测率函数 $b(t)$,当考虑测试工作量时就需要将 $w(t)$包含进来,当从测试覆盖的视角分析时就需要将 $c(t)$融入进来.可以看出,FDR 是对整体测试环境的综合建模,其数学模型不仅反映了故障检测的效率和能力,也描述了当前测试环境下各种随机因素的扰动情况.因此,能否提出能够涵盖真实随机因素的 FDR(整体函数或分段函数)是当前研究面临的一个重要挑战.

6.1.2 发布多模式测试环境下的更多构成要素的失效数据集

包括 FDR 与可靠性在内的各类模型,本质上均需要依靠真实的失效数据集进行验证.现有的失效数据集对 FDR 验证的支持严重不足,这成为制约 FDR 发展的首要障碍.因此,为 FDR 建模提供更多有效信息,为可靠性研究特别是建模、度量、预测、发布、调整等带来重大变革,呼吁公司直接发布 FDR,这也是当前研究中所面临的一个挑战.

6.2 趋势分析

6.2.1 对考虑 FDR 参与的可靠性模型进行综合评价

软件开发自需求分析起始至发布的全过程,包含了多个测试阶段.按照文献[23]中给出的软件开发流程,从图 10 可以看出,为了提高可靠性,四个测试阶段相互衔接直至进入到发布后的运行阶段.

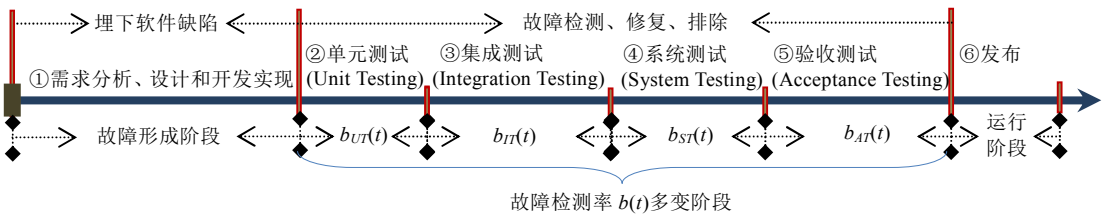


Fig.10 Changes of fault and FDR in software development including multi-test phases

图 10 故障与故障检测率在包含多测试阶段的软件开发过程中的变化

不同测试阶段的目的、策略、技术、方法等差异,使得每个阶段的 FDR 发生变化,这种变化有时较为剧烈,

对测试性能与效率影响较大.FDR 自身的不稳定性对可靠性模型评价带来重要影响,因此要把 FDR 的变化融入到可靠性模型的综合评价分析中,从而为可靠性模型评测带来关键参数上的影响,提高精确度.

6.2.2 FDR 选择成为一个挑战

FDR 支持可靠性研究伴随着其众多模型的提出而发展,特别是在以 SRGM 为核心的可靠性研究上成为关键因素.

易知,提出能够适应多种测试环境变化的 FDR 是建立性能优异的 SRGM 的重要方面.相比之下,现有的 FDS 中却没有公布 FDR 的变化趋势,这使得目前 SRGM 的研究中,尚不能根据 FDS 来直接验证所建立的 $b(t)$,只能间接通过 $m(t)$的性能来体现.

6.2.3 要为最优发布提供有效支持

软件发布受制于软件开发技术、人员、成本等多种因素,通常,超过预期时间的发布因为成本的剧烈上升或失去占领市场的机会等原因而直接导致软件制品的失败.合理管控软件的开发过程,做到在预期之内的发布,甚至是最优发布已成为软件开发管理的重要内容.

区别于传统软件测试的主要目的——单纯的检测与排除故障,当前已开始考虑到持续用以提高可靠性,达到(软件)系统发布的目的.因此,FDR 要能够支撑与服务于软件的及时发布,提高市场占有率.此外,对于新型态软件,例如大规模分布式网络软件、复杂软硬件综合系统、复杂网络软件等,在可靠性研究上缺少失效数据集的外部支持.

6.2.4 基于现有 FDR 选择融入到可靠性模型中,发挥精准组合作用

可靠性建模的一个分支趋势是基于现有的模型框架融入不同的参数,或者分阶段融入不同的参数以得到更为具体的精准模型.FDR 对于建模的重要性还可以通过其作为关键参数的属性,融入到现有的可靠性建模的框架中,进而建立更为灵活的可靠性模型.当前无论是框架模型还是 FDR 模型均存在多个,这为组合方式建立模型提供了有效选择,也为可靠性模型在工程中的应用带来机遇.

6.2.5 根据 FDR 对测试策略实施有效指导

FDR 对测试环境的描述能力直接反映在故障被检测出来的故障数量与效率上,因此有效的 FDR 也应该为实际测试过程给出建议,用以调整测试策略,更加合理地分配测试资源.当前研究中,尚未有从 FDR 的角度对测试工作量（TE）分配、测试过程管理等进行具体的研究,这成为亟待突破的研究内容.

7 结束语

故障检测率 FDR 与可靠性的建模与度量紧密相关,是软件测试过程中测试技术综合运用取得的结果,即可以从测试覆盖的角度进行建模,也可以融合测试工作量 TE 因素,还可以直接根据实际进行设定.可以看出,FDR 是可靠性建模、增长、度量、系统发布的重要构成要素,是用以支撑可靠性研究与增强可靠性增长的重要内容,推动了以 SRGM 为核心的可靠性研究的深入发展.

本文对故障检测率进行了全面述评,包括可靠性建模的关键要素、功能、与失效强度和冒险率的关联、多视角下的分类、不完美排错下的性能分析,以及未来研究趋势等.期望通过我们的工作能为可靠性研究,特别是 FDR 的研究提供有益的借鉴和参考,并为推动可靠性相关的研究与应用向前发展做出积极贡献.

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] 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 with English abstract).http://www.jos.org.cn/1000-9825/ 5306.htm

[2] 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). [doi: 10.3724/SP.J.1016.2014.02586]

[3] 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]

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

[5] Cinque M , Cotroneo D , Pecchia A , et al. Debugging-workflow-aware software reliability growth analysis[J]. Software Testing Verification and Reliability, 2017(1):e1638. [doi: 10.1002/stvr.1638]

[6] Zhang C, Meng FC, Cui G, Liu HW, Guan HY. Overview of modeling of TE in SRGM and comparisons for models. Journal of Harbin Institute of Technology, 2015,47(5):32-39(in Chinese with English abstract). [doi: 10.11918/j.issn.0367-6234.2015.05.006]

[7] Zhang C, Meng FC, Wan K, Chen ZP, Liu HW, Cui G. Analysison SRGM modeling categories and performances. Journal of Harbin Institute of Technology, 2016,48(8):171−178(in Chinese with English abstract). [doi: 10.11918/j.issn.0367-6234.2016.08.029]

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

[9] Liu HW, Yang XZ, Qu F, Dong J. A software reliability growth model with bell shaped fault detection rate function. Chinese Journal of Computers, 2005 (05):908−913(in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2005.05.020]

[10] Song K Y , Chang I H , Pham H . A three-parameter fault-detection software reliability model with the uncertainty of operating environments[J]. Journal of Systems Science and Systems Engineering, 2017, 26(1):121-132. [doi: 10.1007/s11518-016-5322-4]

[11] Pham, Hoang. A generalized fault-detection software reliability model subject to random operating environments[J]. Vietnam Journal of Computer Science, 2016, 3(3):145-150. [doi: 10.1007/s40595-016-0065-1]

[12] Xu JJ, Yao SZ. Characterizing uncertainty of software reliability growth model. Ruan Jian Xue Bao/Journal of Software, 2017,28(7):1746-1758(in Chinese with English abstract). [doi: 10.13328/j.cnki.jos.005108]

[13] Wang JY, Zhang C, Mi XP, Guo XF, Li JH. Software reliability growth model based on Weibull distribution introduced faults. Ruan Jian Xue Bao/Journal of Software, 2019,30(6):1759-1777(in Chinese with English abstract). [doi: 10.13328/j.cnki.jos.005427]

[14] Wang J, Zhang C. Software Reliability Prediction Using a Deep Learning Model based on the RNN Encoder–Decoder[J]. Reliability Engineering & System Safety, 2018, 170, February 2018, Pages 73-82(https://doi.org/10.1016/j.ress.2017.10.019)

[15] Huang C, Lyu M. 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]

[16] Mengmeng Zhu, Hoang Pham. A software reliability model with time-dependent fault detection and fault removal[J]. Vietnam Journal of Computer Science, 2016, 3(2):71-79. [doi: 10.1007/s40595-016-0058-0]

[17] Chiu K C, Huang Y S, Lee T Z. A study of software reliability growth from the perspective of learning effects[J]. Reliability Engineering & System Safety, 2008, 93(10):1410-1421.[doi: 10.1016/j.ress.2007.11.004]

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

[19] 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]

[20] 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]

[21] 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]

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

[23] Rana R, Staron M, Berger C, et al. Selecting software reliability growth models and improving their predictive accuracy using historical projects data[J]. Journal of Systems & Software, 2014:59-78.[doi: 10.1016/j.jss.2014.08.033]

[24] 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].

[25] Princy BA, Sridhar S. Measuring software reliability and release time using SRGM tool. International Journal Of Scientific Research And Education, 2014,2(05):785-796.

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

[27] Pham H. Software reliability and cost models: Perspectives, comparison, and practice[J]. European Journal of Operational Research, 2003, volume 149(3):475-489. [doi: 10.1016/S0377-2217(02)00498-8]

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

[29] Kapur P K, Goswami D N, Bardhan A, et al. Flexible software reliability growth model with testing effort dependent learning process[J]. Applied Mathematical Modelling, 2008, 32(7):1298-1307.[doi: 10.1016/j.apm.2007.04.002]

[30] Xu RZ. Software Reliability Engineering, 1st ed. Beijing, China:Tsinghua University Press. 2017. 18-20.

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

[32] 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].

[33] Pham, Hoang. A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments[J]. Optimization, 2014, 63(10):1481-1490. [doi: 10.1080/02331934.2013.854787]

[34] 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

[35] Kapur P K, Anand S, Singh V B. Distribution Based Change-Point Problem With Two Types of Imperfect Debugging in Software Reliability[J]. Bvicams International Journal of Information Technology, 2002, 1(2).

[36] 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].

[37] 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. International Journal of Production Research, 2008,46(3):771-787. [doi: 10.1080/00207540600926113].

[38] Yadavalli V S S, Aggarwal A G, Kapur P K, et al.. Unified framework for developing testing effort dependent software reliability growth models with change point and imperfect debugging[C]. Proc of the 4th National Conf on Computing For Nation Development. New Delhi, 2010: 509-516.

[39] 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].

[40] 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].

[41] 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. International Journal of Operational Research, 2014,21(2):201-220. [doi: 10.1504/IJOR.2014.064544].

[42] Pachauri B , Dhar J , Kumar A . Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth[J]. Applied Mathematical Modelling, 2015, 39(5-6):1463-1469. [doi: 10.1016/j.apm.2014.08.006]

[43] Anu Gupta Aggarwal, Neha Gandhi, Vibha Verma , et al. Multi-release software reliability growth assessment: an approach incorporating fault reduction factor and imperfect debugging[J]. International Journal Mathematics in Operational Research, 2019, 15(4): 446-463. [doi: 10.1504/IJMOR.2019.10016194]

[44] 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. [doi: 10.1109/tc.2009.103]

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

[46] Li Q , Pham H . NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage[J]. Applied Mathematical Modelling, 2017, 51:68-85. [doi: 10.1016/j.apm.2017.06.034]

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

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

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

[50] Gokhale S S, Lyu M R, Trivedi K S. Analysis of Software Fault Removal Policies Using a Non-Homogeneous Continuous Time Markov Chain[J]. Software Quality Control, 2004, 12(3):211-230. [doi: 10.1023/B:SQJO.0000034709.63615.8b]

[51] Bergel A , Peña V . Increasing test coverage with Hapao[J]. Science of Computer Programming, 2014, 79:86-100.[doi: 10.1016/j.scico.2012.04.006]

[52] Gokhale S S, Philip T, Marinos P N, et al. Unification of finite failure non-homogeneous Poisson process models through test coverage[J]. Proc Seventh International Symposium on Software Reliability Engineering White Plains, 1996:299-307.[doi: 10.1109/ISSRE.1996.558886]

[53] Anni Princy B, Sridhar S. An efficient software reliability growth models with two types of imperfect debugging[J]. European Journal of Scientific Research, 2012, 72(4): 490-503.

[54] Gokhale S S, Trivedi K S. A time/structure based software reliability model[J]. Annals of Software Engineering, 1999, 8(1):85-121. [doi: 10.1023/A:1018923329647]

[55] Malaiya Y K, Li M N, Bieman J M, et al. Software reliability growth with test coverage[J]. Reliability IEEE Transactions on, 2002, 51(4):420-426.[doi: 10.1109/tr.2002.804489]

[56] 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]

[57] Li QY, Li HF, Lu MY. Software reliability growth model with S-shaped testing effort function. Journal of Beijing University of Aeronautics and Astronautics, 2011,37(2):149−154(in Chinese with English abstract).[doi: 10.13700/j.bh.1001-5965.2011.02.025]

[58] 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).

[59] Li HF, Wang XC, Li QY. Software Reliability Growth Model Considering Imperfect Debugging with Logistic Testing Coverage Function. Journal of Computer Research and Development, 2010,47(Suppl.):216−223(in Chinese with English abstract).

[60] Li HF, Wang SQ, Liu C, Zheng J, Li Z. Software reliability model considering both testing effort and testing coverage. Ruanjian Xuebao/Journal of Software, 2013,24(4):749-760(in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4257.htm

[61] Chen M H, Lyu M R, Wong W E. Effect of code coverage on software reliability measurement[J]. Reliability IEEE Transactions on, 2001, 50(2):165-170.

[62] SHINJI INOUE, SHIGERU YAMADA. TESTING-COVERAGE DEPENDENT SOFTWARE RELIABILITY GROWTH MODELING[J]. International Journal of Reliability Quality & Safety Engineering, 2011, 11(4):303-312.[doi: 10.1142/S0218539304001531]

[63] Cai X, Lyu M R. Software Reliability Modeling with Test Coverage: Experimentation and Measurement with A Fault-Tolerant Software Project[C]// 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE). IEEE Computer Society, 2007:17-26.[doi: 10.1109/ISSRE.2007.17]

[64] 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].

[65] Li QY, Li HF, Lu MY. Incorporating S-shaped testing-effort functions into NHPP software reliability model with imperfect debugging[J]. Journal of Systems Engineering and Electronics,2015,26(01):190-207. [doi: 10.1109/JSEE.2015.00024]

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

[67] 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].

[68] 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].

[69] 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.[doi: 10.1016/j.jss.2005.06.039]

[70] Ahmad N, Bokhari M U, Quadri S M K, et al. The exponentiated Weibull software reliability growth model with various testing-efforts and optimal release policy[J]. International Journal of Quality & Reliability Management, 2008, 25(2): 211-235.

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

[72] 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].

[73] Ahmada N, Khanb M G M, Rafib L S. Analysis of an inflection S-shaped software reliability model considering log-logistic testing-effort and imperfect debugging[J]. Int. J. Comp. Net. Sec, 2011, 11(1): 161-170.

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

[75] 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.

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

[77] Musa J D , Iannino A , Okumoto K . Software Reliability: Measurement, Prediction, Application[M]. McGraw-Hill, 1990.

[78] Pham. H, Software reliability[M]. Singapore: Springer, 2000.

[79] 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]

[80] Misra, P. N. Software reliability analysis, IBM Journal of Research Development, 1984, 22. 262-270.

[81] 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].

[82] 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].

[83] 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].

[84] 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). [doi: 10.3724/SP.J.1001.2010.03539]

附中文参考文献:

[1] 张策,孟凡超,考永贵,刘宏伟,崔刚,吕为工.软件可靠性增长模型研究综述.软件学报,2017,28(9):2402-2430.

[2] 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.

[6] 张策,孟凡超,崔刚,刘宏伟,官红玉.SRGM 中 TE 建模机制与模型比较分析.哈尔滨工业大学学报,2015,47(5):32-39.

[7] 张策,孟凡超,万锟,陈智朋,刘宏伟,崔刚.SRGM 建模类别与性能分析.哈尔滨工业大学学报,2016,48(8):171-178.

[9] 刘宏伟,杨孝宗,曲峰,董剑.一个基于响铃形故障检测率函数的软件可靠性增长模型.计算机学报,2005(05):908-913.

[12] 许家俊,姚淑珍.软件可靠性增长模型的不确定性量化研究.软件学报,2017,28(7):1746-1758[doi:10.13328/j.cnki.jos.005108]

[13] 王金勇,张策,米晓萍,郭新峰,李济洪.Weibull 分布引进故障的软件可靠性增长模型.软件学报,2019,30(6):1759-1777.

[30] 徐仁佐.软件可靠性工程.第一版,北京:清华大学出版社,2007.18-20.

[34] 王金勇,吴智博,舒燕君,等.故障检测率不规则变化的软件可靠性模型.软件学报,2015,26(10):2465-2484.

[56] 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.

[57] 李秋英,李海峰,陆民燕,等.基于 *S* 型测试工作量函数的软件可靠性增长模型.北京航空航天大学学报,2011,37(2):149-154.

[58] 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.

[59] 李海峰,王学成,李秋英,等.考虑不完美排错的 Logistic 测试覆盖率软件可靠性模型.计算机研究与发展,2010,47(Suppl.): 216-223.

[60] 李海峰,王栓奇,刘畅,等.考虑测试工作量与覆盖率的软件可靠性模型.软件学报,2013,(4):749-760.

[84] 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.