软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(10):2465−2484 [doi: 10.13328/j.cnki.jos.004746]

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

故障检测率不规则变化的软件可靠性模型*

王金勇, 吴智博, 舒燕君, 张 展

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

通讯作者: 张展, E-mail: zhangzhan@hit.edu.cn; 王金勇, E-mail: wangjinyong818@163.com

摘 要: 传统的NHPP(non-homogeneous Poisson process)模型在实际的测试当中被证明是成功的.但是,由于传统的NHPP模型用的是理想的假设,例如,假设故障检测率是常数、平稳变化和规律变化,模型的性能在实际的测试环境中总是受到损害.因此,提出一个基于NHPP的软件可靠增长模型,并且考虑故障检测率的不规则变化情况,这种变化符合故障检测率在实际的软件测试过程中的变化.通过相关的实验验证了所提出的 NHPP 模型的拟合和预测能力.实验结果表明:在用实际的故障数据进行拟合和预测的过程中,所提出的模型与传统的 NHPP 模型相比,有更好的拟合和预测性能.同时,也给出了所提出模型相应的置信区间.

关键词: 软件可靠性增长模型;非齐次泊松过程;不规则变化;故障检测率;软件可靠性

中图法分类号: TP311

中文引用格式: 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,2015,26(10):2465−2484. http://www.jos.org.cn/1000-9825/4746.htm

英文引用格式: 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). http://www.jos.org.cn/1000-9825/4746.htm

Software Reliability Model with Irregular Changes of Fault Detection Rate

WANG Jin-Yong, WU Zhi-Bo, SHU Yan-Jun, ZHANG Zhan

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

Abstract: The traditional NHPP (non-homogeneous Poisson process) models are proved to be a success in a practical test. However, the model performance always suffers in the realistic software testing environment due to the ideal assumption which derived the traditional NHPP models, such as constant fault detection rate and smooth or regular changes. In this paper, an NHPP-based software reliability growth model is proposed considering an irregular fluctuation of a fault detection rate, which is more in line with the actual software testing process. The fitting and predictive power of the proposed model is validated using the related experiments. The experimental results show the proposed model has a better fitting and predicting performance than the traditional NHPP-based models using the real-world fault data. Meanwhile, the confidence interval is given for the confidence analyses of the proposed model.

Key words: software reliability growth model (SRGM); non-homogeneous Poisson process (NHPP); irregular change; fault detection rate; software reliability

近些年来,计算机应用渗透到了社会和人们生活的各个方面,例如交通系统、银行管理系统和国家防御系统等;它们不但改善了人们的生活品质,而且也提高了人们的工作效率.由于现有计算机硬件技术的不断创新,计算机所发生的故障主要集中在软件方面.一旦计算机软件发生故障,将会对人们的生活、工作,甚至生命造成重大损失.因此,现在计算机工业的主要问题是如何开发高质量的软件系统以及怎样评估软件的可靠性.当软件测试进行到一定阶段,比如验收测试阶段,完全可以利用已产生的故障数据来进行软件可靠性模型拟合和参数

* 基金项目: 国家自然科学基金(61202091, 61173020); 国家高技术研究发展计划(863)(2013AA01A215); 中央高校基本科研业务费专项资金(HIT.NSRIF.2014067)

收稿时间: 2014-04-21; 修改时间: 2014-07-16; 定稿时间: 2014-10-22

估计,然后用软件可靠性模型来预测软件中剩余故障的数量,并进行相应的软件可靠性分析.如果软件可靠性未达到要求,则继续进行相应的软件测试;如果软件可靠性已达到要求,则可以考虑停止该阶段测试,进入下一阶段.实际研究结果表明:建立适当的软件可靠性增长模型,可更加有效和准确地描述软件测试过程中软件故障发生行为,评估并量化软件在测试过程中的可靠性,对及时发布预期要求的软件可靠产品具有重要的指导意义.由于软件测试是一个复杂的问题,受到许多因素的影响.客观因素有:测试的工具、测试的资源、测试环境等;主观因素有:测试者的技术和心理因素等;而且现在已开发的软件可靠性模型在一定条件下是有效和准确的,但在其他条件下则不适用,因此不具有通用性.原因与其故障检测率的理想假设有关,对于预测实际软件故障检测不确定事件来说,它仍是一个没有完全解决的开放性问题.

软件可靠性被定义为:在规定时间和条件下,软件运行不发生故障的概率[1].软件可靠性在量化软件质量方面是一个重要的因素.在预定义的可靠性目标下,可以确定什么时候停止测试,并发布新版本的软件产品.在过去的 40 年里,许多软件可靠性增长模型(SRGM)[2−11]已被开发出来,其中,非齐次泊松过程(non-homogeneous Poisson process,简称 NHPP)类模型在实际的测试中取得了很大成功,也吸引了大量的研究人员加以关注[1].

根据故障检测率的不同变化,NHPP 的软件可靠性增长模型一般可以划分为 3 类:一是故障检测率是不变的,也就是常数;二是故障检测率为平稳变化;三是故障检测率是有规律变化的.

- 对于第 1 类软件可靠性模型,Goel[3]假设:在$(t,t+\Delta t)$时间内,期望检测出的故障数量和在 t 时间内软件中剩余故障的数量成正比,故障检测率是一个比例常数.这个模型也称为指数软件可靠性增长模型.为了描述故障强度在测试开始时增加并且随后下降的趋势,Goel[12]提出了一个一般化的软件可靠性模型(GGO).除了完美调试模型,即:假设在检测到的故障立即被去除,没有新的故障被引进外,Goel[12]还首次提出了不完美调试模型,即:假设在去除故障时,有新的故障被引进,并开发了故障检测率为常数的不完美软件调试模型;
- 对于第 2 类软件可靠性模型——故障检测率为平稳变化的模型,Yamada 等人[3]观察到:在软件测试过程中,累计检测到的故障形状为 S 型,因此提出了 Delay S-shaped 模型(DSS).另一个相似的模型为 Inflection S-shaped 模型(ISS)[4],它描述了在软件测试当中,软件故障之间存在故障依赖现象.在日本,许多软件开发商采用 Logistic 和 Gompertz 增长曲线模型[5]来评测软件的可靠性,因为它们是最简单的 S-shaped 可靠性模型.在不完美调试模型当中,也有许多有关平稳变化的故障检测率模型被开发出来[13−16];
- 但在实际的测试过程中,故障检测率也许不是平稳变化的,可能会发生规律变化.这就是提到的第 3 类软件可靠性模型,也就是故障检测率为有规律变化的情况.已经有许多文章研究了这种变化[17−20],如 Yamada 等人[17]考虑了测试工作量(test effort)的变化,并且研究了测试工作量和故障检测数量之间的关系,提出了一个有关测试工作量的模型.另一方面,在实际测试过程中,软件测试过程是复杂的并且受到很多因素的影响,例如测试资源的分配、测试策略的使用以及测试环境的变化等,因此,故障检测率也许不是平稳变化的,可能在某个时间点上发生变化,这也被称为移动点[21−27].例如:Huang 等人[27]考虑了故障检测和去除过程的不同,他们用统一理论并合并了多移动点的概念,将其带入软件可靠性模型.

通过上面的分析我们可以合理地得出故障检测率的 3 种变化:一是常数,例如 G-O 模型;二是平稳变化,例如 DSS 模型;三是有规律的变化,例如测试工作量和移动点模型.但是,这些模型在实际测试中,它们的故障拟合和预测性能都会受到损害.因为故障检测是一个复杂和随机的过程,它是一个不确定事件,也就是一个概率事件.在实际故障检测过程中,故障检测率随着测试环境、测试人员和测试资源的变化而发生变化,而且发生的是未知突然变化,即不规则变化.图 1 给出了故障检测率随测试时间不规则变化的情况.从图 1 中我们能够清楚地看到,在软件测试中,故障检测率随测试时间发生不规则的上下变化情况.因此在本文中,我们开发了一个在实际测试过程中随时间不规则变化的 NHPP 软件可靠性增长模型,目的是为了更深入地研究软件测试过程中故障检测率不规则变化的现象,用新方法来提高软件可靠性模型的拟合和预测性能,进而使建立的软件可靠性模型更接近于实际的软件测试过程变化.实验结果显示:我们提出的模型有更好的拟合历史故障数据集的能力,并能

提供更精准的软件剩余故障数量的预测.同时,我们提出的模型在适应各种测试环境方面也有很强的鲁棒性(robust).

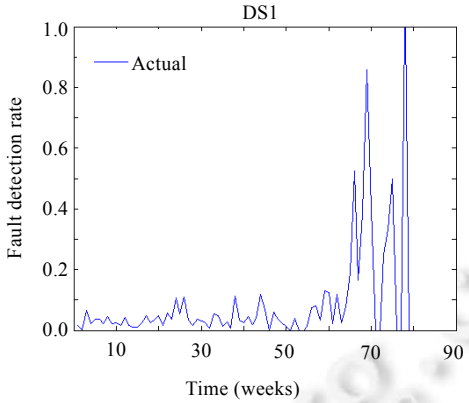

Fig.1 Fault detection rate irregularly changes with the testing time

图 1 故障检测率随测试时间不规则变化的情况

1 软件可靠性增长模型开发

- 缩略语
  - K-S:Kolmogorov-Smirnov
  - KD:Kolmogorov-Distance
  - TEF:测试工作量函数
  - CP:移动点
  - PM:本文提出的模型
- 符号说明
  - a:期望最初故障数量
  - b:故障检测率
  - b_i:在移动点模型中的第 i 个故障检测率
  - τ:在移动点中的时间点
  - k:故障样本的数量
  - φ:故障引进率
  - μ:模型参数数量
  - c:常量参数
  - β:拐点率
  - δ:波动因子
  - r:威布尔类测试工作量(Weibull-type TEF)模型中比例参数
  - o_{t_i} :到时刻 t_i 为止,观测到的故障数量
  - A:威布尔类测试工作量(Weibull-type TEF)模型中常量参数
  - N:测试过程中,总共花费的测试工作量
  - $b(t)$:故障检测率函数
  - $N(t)$:随机变量表示到 t 时刻为止,实际检测出的故障数量
  - $m(t_i)$:均值函数,即:在$[0,t_i]$时间内,检测到的累计的故障数量期望值;$m(t_i)=E[N(t_i)]$

- $W(t)$:在 t 时刻,累计消耗的测试工作量
- $F^*(x)$:到时刻 x 为止,观察到的标准化的累计分布
- $F(x)$:到时刻 x 为止,期望的累计分布

1.1 基本假设

在本文中,PM 的提出基于以下假设:

① 故障检测服从 NHPP 过程;

② 最初软件系统的故障是一个随机变量;

③ 软件可以在任意时间发生故障,且由软件中剩余故障造成;

④ 在故障检测过程中,每次只有一个故障被发现,检测出的故障立即被去除,没有引进新的故障;

⑤ 故障检测率是一个随时间发生不规则变化的随机变量;

⑥ 发生故障(k)的时间和发生故障(k–1)的时间有关.

假设⑤捕获了在实际测试过程中故障检测率的变化特征,假设⑥说明故障之间有依赖关系,由假设③可以建立微分等式方程.

1.2 模型开发

一般来说,假设$\{N(t),t>0\}$表示故障计数过程,则 NHPP 过程可以表示为

$$\Pr\{N(t)=k\}=\frac{(m(t))^K}{k!}\mathrm{e}^{-m(t)},k=0,1,2,... \tag{1}$$

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

其中,$m(t)$为均值函数,$\lambda(x)$为故障强度函数.因此,不同假设的 $m(t)$会有不同的软件可靠性模型.一般由假设③和假设⑤可建立以下微分方程:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b(t)(a-m(t)) \tag{3}$$

$$b(t)=bt^d+\delta\gamma(t) \tag{4}$$

其中,$b(t)$是故障检测率函数;b 为故障检测率;a 为最初软件中期望的故障数量;d 是一个塑形参数;$\gamma(t)$是一个高斯白色噪音(Gaussian growth noise);δ是一个正常数,表示不规则波动过程.

把等式(4)带入等式(3)可得:

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=(bt^d+\delta\gamma(t))(a-m(t)) \tag{5}$$

把等式(5)扩展到伊藤类(Itô type)[28,29]的微分等式,可得:

$$\mathrm{d}m(t)=\left(bt^d-\frac{1}{2}\delta^2\right)m(t)\mathrm{d}t+\delta m(t)\mathrm{d}\xi(t) \tag{6}$$

其中,$\xi(t)$是一维威纳(Wiener)过程,被形式化定义为白色噪音$\gamma(t)$关于时间 t 的积分.威纳过程是一个高斯(Gaussian)过程,并具有下列属性:

$$\Pr[\xi(0)=0]=1 \tag{7}$$

$$E[\xi(t)]=0 \tag{8}$$

$$E[\xi(t)\xi(t')]=\mathrm{Min}[t,t'] \tag{9}$$

威纳过程和布朗运动(Brownian motion)服从正态分布,并且$\xi(t)$的密度函数为

$$f(\varsigma(t))=\frac{1}{\sqrt{2\pi t}}\exp\left\{-\frac{(\varsigma(t))^2}{2t}\right\} \tag{10}$$

利用伊藤公式[28,29]和等式(10),当 $m(0)=0$ 时,等式(5)可解得:

$$\begin{aligned} m(t) &= a(1-\exp(-\int_0^t b(u)\mathrm{d}u)) \\ &= a(1-\exp(-\int_0^t (bu^d+\delta\gamma(u))\mathrm{d}u)) \\ &= a(1-\exp(-(bt^{(d+1)}/(d+1)+\delta\xi(t)))) \\ &= a(1-\exp(-(bt^{(d+1)}/(d+1)-\delta^2 t/2))) \end{aligned} \tag{11}$$

当 t=0 时,$m(0)$=0;当 $t\to\infty$时,$m(\infty)=a$.

- 故障强度函数可表示为

$$\lambda(t)=\frac{\mathrm{d}m(t)}{\mathrm{d}t}=a(bt^d-\delta^2/2)\exp(-(bt^{(d+1)}/(d+1)-\delta^2 t/2)) \tag{12}$$

- 故障检测率函数可表示为

$$b(t)=\frac{\lambda(t)}{a-m(t)}=bt^d-\delta^2/2 \tag{13}$$

在等式(11)中:参数 a 表示在软件测试过程中,期望检测出故障的数量,可用 a 来估计软件中存在的故障总数量;参数 b 表示在软件测试过程中,检测到故障的发生率,可以用 b 大致估计软件测试中检测到故障的效率;参数 d 为塑型参数,它可以非线性地表示故障检测发生的情况;δ为波动因子,它可以表示故障检测率不规则波动程度.因此,PM 的 4 个参数对于准确估计软件可靠性具有重要意义.

2 相关工作

为了说明在传统的基于 NHPP 模型中故障检测率开发的多样性,我们在这一节回顾一些相关的 NHPP 模型的开发过程.

在过去的 40 年里,研究者们提出了许多软件可靠性增长模型.软件可靠性增长模型假设软件的失效是由软件中存在的故障造成,并且是随机发生的;当每次发生故障时,它会被立即去除,并且没有新的故障被引进.其中,软件可靠性增长模型中很重要的一个类型是非齐次泊松过程(NHPP),它在实际的软件测试中取得了很大成功,并吸引了大量人员去研究它[1].首先,Goel 和 Okumoto[2]提出了一种简单和成功的基于 NHPP 的模型,这个模型也被称为指数分布的软件可靠性增长模型.他们假设:软件测试中,期望的故障检测的数量正比于软件测试中软件故障剩余的数量,并且故障检测率为常数.为了描述软件故障检测时软件故障强度(fault intensity)在开始时略微增加、在结束时下降的趋势,Goel 还提出了另一个带有 3 个参数的一般的 GGO 模型[12],它也假设故障检测率为常数.

虽然假设常量故障检测率能够简化建立软件可靠性增长模型的过程,但是由于软件测试是一个复杂的过程,假设故障检测率为常数不能动态且准确地反映软件测试过程中故障检测现象.因此,Yamada 等人[3]在研究累计的软件故障检测数量后,发现增长的曲线为 S 型,提出了 Delay S-shaped 软件可靠性增长模型.Ohba[4]发现软件测试过程中软件测试者的学习现象后,提出了另一种 Inflection S-shaped 软件可靠性增长模型.以上两种 S-shaped 模型都是假设故障检测率为平稳变化的增长型函数.另外,Yamada 等人[17]考虑了测试工作量(test effort)的变化,并且研究了测试工作量和故障检测数量之间的关系,提出了一个有关故障检测率为平稳变化测试工作量的模型.

虽然假设故障检测率为平稳变化的增长模型在一定程度上能够有效和准确地评估和预测在测试中软件故障的数量,但是由于在实际的测试过程中软件测试过程可能受许多因素影响,例如测试资源的分配、测试策略和测试运行环境等,因此软件故障检测率可能不是平稳变化的,而有可能是有规律变化的.例如,在软件测试过程中,可能故障检测率在某个时间发生变化,则称为移动点(changing-point)[21−27].Huang 等人[27]考虑到:在软件开发和运行过程中,他们用统一理论并且合并多个移动点来建立基于 NHPP 的软件可靠性增长模型,并充分研究了故障检测率在测试过程中有规律变化的现象.

除了上面提到的完美调试模型,假设检测到的故障立即去除没有新的故障被引进外,另一种不完美调试模

型也被许多学者开发出来[12,14−16,23],即:假设检测到的故障被去除时,又有新的故障可能产生.他们也开发了许多像完美软件调试模型那样假设故障检测率为常数、平稳变化的和有规律变化的不完美调试模型.例如:

- Goel[12]首次提出了不完美调试模型,即:假设在去除故障时,有新的故障被引进,并开发了故障检测率为常数的不完美软件调试模型;
- Huang等人[30]考虑故障检测和纠错的过程时间延迟过程,他们提出了有限和无限服务排队理论的基于NHPP不完美软件可靠性增长模型.Huang等人[27]也采用多移动点的方法来解决不完美调试问题;
- Xiao和Dohi[31]则用小波变换的方法来量化评估NHPP类的软件可靠性模型的性能;
- Wang等人[32]则用概率状态转移来研究完美调试和不完美调试在软件测试中的相互关系.

从上面的讨论我们可以合理地得出:假设软件故障检测率为常数、平稳变化和有规律变化,能够在一些软件测试情况下,有效和准确地评估软件的可靠性.但在实际的软件测试过程中,软件测试不但受到客观因素影响,例如软件测试的工具等,还受到主观因素的影响,例如软件测试者的技能和心理等.因此,软件测试是一个复杂和非确定事件,在实际的软件测试过程中,软件故障检测率最有可能表现为无规律的不规则变化.研究在软件测试过程中故障检测率为不规则变化的NHPP类软件可靠性增长模型,更符合实际的软件故障检测过程,并具有更重要的实际意义.本文提出了一种考虑故障检测率不规则变化的NHPP类的软件可靠性增长模型,并用8个历史故障数据集进行了相应的验证和评估.实验结果表明:与其他NHPP模型相比,PM有更好的拟合和预测性能.

3 数值例子

在这一节,PM将与其他不同条件下开发的NHPP模型进行充分的比较.表1给出了所有的模型和均值函数(MVF).同时,为了说明PM的可信性(confidence),也进行了相关的置信区间的估计和分析.此外,所有模型的参数是用最大似然估计(MLE)方法估计出来的[7,33−35].最大似然估计(MLE)方法估计模型参数,与其他估计模型参数的方法相比具有无偏性的优点.而其他方法,比如最小二乘法没有这个优点,如果在大样本的情况下,用它进行模型参数估计会对结果造成一定的影响.

Table 1 Various NHPP models and proposed models

表1 不同的NHPP模型和PM

模型	均值函数(MVF)
Goel-Okumoto (G-O)[2]	$a(1-\exp(-bt))$
Generalized Goel model (GGO)[16]	$a(1-\exp(-bt^c))$
Delay S-shaped (DSS)[3]	$a(1-(1+bt)\exp(-bt))$
Inflection S-shaped (ISS)[4]	$\frac{a(1-\exp(-bt))}{1+\varphi\exp(-bt)}$
G-O with logistic TEF (GOLT)[20]	$a(1-\exp(-b(W(t)-W(0)))),W(t)=\frac{N}{1+A\exp(-\alpha t)}$
Delay S-shaped model with logistic TEF (DSSMLT)[20]	$a(1-(1+b(W(t)-W(0)))\exp(-b(W(t)-W(0)))),W(t)=\frac{N}{1+A\exp(-\alpha t)}$
G-O model with a single CP (GOMSC)[27]	$a(1-\exp(-(b_1\tau+b_2(t-\tau))))$
Inflection S-shaped model with a single CP(ISSMSC)[27]	$\frac{a(1-\exp(-(b_1\tau+b_2(t-\tau))))}{1+\beta\exp(-(b_1\tau+b_2(t-\tau)))}$
Proposed new model (PM)	$a(1-\exp(-(bt^{(d+1)}/(d+1)-\delta^2 t/2)))$

3.1 最大似然估计(MLE)

在这一节,我们给出等式(11)中未知参数a,b,d和δ的估计方法,而它们的联合概率函数可以表示为

$$P(t_1,y_1;t_2,y_2;\ldots;t_n,y_n)=\Pr\{m(t_1)\leqslant y_1,\ldots,m(t_n)\leqslant y_n|m(t_0)=0\} \quad (14)$$

其中,$m(t)$表示软件测试时间在t时刻累计检测到的故障数量,其概率密度函数可以表示为

$$p(t_1,y_1;t_2,y_2;...;t_n,y_n)=\frac{\partial^n P(t_1,y_1;t_2,y_2;...;t_n,y_n)}{\partial y_1\partial y_2...\partial y_n} \tag{15}$$

因为 $m(t)$可以表示为连续值,其似然函数 l 可以表示为

$$l=p(t_1,y_1;t_2,y_2;\ldots;t_n,y_n) \tag{16}$$

我们可以进一步简化计算,而采用对数似然函数形式:

$$L=\log l \tag{17}$$

通过联立解以下方程,可以得出最大似然估计值 a^*,b^*,d^*和δ^*.

$$\frac{\partial L}{\partial a}=\frac{\partial L}{\partial b}=\frac{\partial L}{\partial d}=\frac{\partial L}{\partial \delta}=0 \tag{18}$$

3.2 故障数据集

为了充分和有效地验证所提出模型的性能,我们选用 8 个故障数据集进行测试,其中,前 3 个故障数据集用于部分模型的性能比较,后 5 个故障数据集用于全部模型的性能比较,并验证了 PM 在实际软件测试中的性能.

- 第 1 个故障数据集来自巴西电气转化系统(Brazilian electronic switching system)——TROPICO-R 1500,它是面向 1 500 个电话客户的系统[36].这个故障数据集有 81 个数据条目,大小为 300KB.测试用了 81 周的时间,461 故障被去除.需要注意的是:从第 1 周~第 30 周的数据是在系统验证阶段获得的,从第 31 周~第 42 周的数据是在域测试阶段获得的,第 42 周~第 81 周的数据是在系统操作阶段获得的;
- 第 2 个故障数据集来自无线网络产品[37],它是运行在无线网络转换中心的软件.测试用了 51 周的时间,203 个故障被去除;
- 第 3 个数据集是来自 Ohba 的文章,它是 PL/2 数据库应用软件系统,包括 1 317 000 LOC[38].用了 19 周的时间,花费了 47.65CPU 小时,328 故障被去除;

第 4 个故障数据集~第 8 个故障数据集是 Musa[39]收集并整理的实际软件测试过程产生的不同类型的故障数据.其中,

- 第 4 个故障数据集来自实时命令和控制系统(real time command and control),系统码(system code)为 1,交付的目标代码指令数(delivered object code instructions)为 21 700,并且在 92 天里,总共花费了 88 682 个执行时间(executive time),检测并去除了 136 个故障;
- 收集的第 5 个故障数据集产生于操作系统(operating system),系统码是 SS1A,交付的目标代码指令数为几十万,用了 148 天,总共花费 16 501 830 个执行时间,检测并去除了 112 个故障;
- 第 6 个故障数据集来自于时间分享系统(time sharing system),系统码是 SS2,交付的目标代码指令数为几十万,在 655 天里,共花费 60 262 806 个执行时间,检测并去除了 192 个故障;
- 第 7 个故障数据集产生于词处理系统(word processing system),系统码是 SS3,交付的目标代码指令数为几十万,并且在 657 天里,共花费了 57 676 499 个执行时间,检测并去除了 278 个故障;
- 第 8 个故障数据集产生于操作系统,系统码是 SS4,交付的目标代码指令数为几十万,并且在 619 天里,共花费了 49 239 302 个执行时间,检测并去除了 196 个故障.

需要注意的是:我们选择的验证模型性能的故障数据集是有特点的,其中,

- 前 3 个历史故障数据集用于部分模型在特定测试环境中的性能比较.例如:第 1 个故障数据集(DS1)可用于移动点模型验证,第 2 个故障数据集(DS2)可用于传统的 NHPP 模型验证,第 3 个故障数据集可用于测试工作量模型的验证;
- 后 5 个故障数据集用于全部模型的性能比较,并且用 Musa 在实际的软件测试中收集的有代表性的不同软件可靠性故障数据集来验证我们提出的软件可靠性模型在实际软件测试过程中的性能,包括预测和拟合软件故障数量的有效性和准确性.其中,选用的比较模型包括传统的 NHPP 模型、移动点模型和测试工作量模型.

3.3 模型比较标准

为了比较模型的故障拟合和预测能力,我们采用了 3 个模型评比标准.

标准 1.

均值平方误差(MSE)用来描述累计故障数和实际预测值之间的平均距离,其定义为

$$MSE=\frac{\sum_{i=1}^{k}[m(t_i)-o_{t_i}]^2}{k} \tag{19}$$

其中,k表示故障数据集的故障样本的数量,o_{t_i}表示到时刻t_i观测到的故障数,$m(t_i)$表示到时刻t_i模型估计得到的故障数.均值平方误差越小,说明评比模型故障拟合或预测的性能越好.

一般来说,模型的参数越多,模型故障拟合或预测的性能越好.为了减少模型参数对模型性能的影响,我们给出了第 1 个评比标准的变化形式,其定义为

$$MSE_1=\frac{\sum_{i=1}^{k}[m(t_i)-o_{t_i}]^2}{k-\mu} \tag{20}$$

其中,μ表示模型参数的数量.其他参数同上.

标准 2.

第 2 个评比标准为 K-S 测试,一般认为,K-S 测试比卡方测试(chi-square)更能获得评比信息[33,40].K-S 测试可以量化模型累计分布函数和样本分布函数的距离,其定义为

$$D_h=\mathrm{Sup}_x|F^*(x)-F(x)| \tag{21}$$

其中,h 是样本的大小.D_h用于测量经验和估计分布函数的接近程度,换句话说,它是故障观测率和估计故障期望率的绝对差值.我们可以选择故障数量差值最大的绝对值作为 KD 距离[27].越小的 KD 值,模型故障拟合或预测的性能越好.

标准 3.

第 3 个模型的比较标准是 AIC(the Akaike information criterion)[41],它用于计算模型在拟合故障数据时估计模型参数的似然函数达到最大值的能力.一般来说,模型的参数越多,模型的拟合性能越好.而 AIC 比较标准则加入了对模型的参数惩罚,也就是说,模型的参数越多,AIC 值会越大,模型的拟合性能会越差.其定义如下:

$$AIC=-2\log(\text{似然函数取最大值})+2N \tag{22}$$

其中,N 表示模型中参数的数量.越小的 AIC 值,模型的拟合性能越好.

3.4 模型比较和分析

在这一节,为了验证所提出模型在实际测试中的有效性和预测软件测试中故障数量的准确性,我们使用了多种数据集.除使用了上面介绍的故障数据集外,还使用了 Musa[39]收集并整理的实际测试中的软件可靠性数据集(the software reliability dataset),该数据集被广泛用于软件可靠性模型的验证和评估.我们还进行了相关模型的故障拟合和预测性能比较,用到的模型包括 PM、传统的 NHPP、移动点以及测试工作量等模型.

3.4.1 部分模型拟合和预测性能比较

1) 数据集 1(DS1)

从表 2 可以看到:PM 在与其他模型的比较过程中,拟合 MSE_{1fit} 和 KD_{fit} 以及预测 $MSE_{1predict}$ 和 $KD_{predict}$ 值都是最小的,它们分别是 90.1,0.057 6 和 336.5,0.072;第二小的是 Inflection S-shaped model with a single CP (ISSMSC),它们分别是 96.6,0.061 1 和 437.5,0.098.拟合 MSE_{1fit} 和 KD_{fit} 以及预测 $MSE_{1predict}$ 和 $KD_{predict}$ 值都为最大的是 Delay S-shaped(DSS),它们分别是 250.1,0.089 4 和 1 627.3,0.16.因此,PM 拟合及预测的 MSE_1 和 KD 值都少于 G-O、DSS、单移动点 G-O 和单移动点 ISS 模型,这说明 PM 在与其他模型相比时有很强的故障拟合和预测能力.这些结果也可从图 2(a)中看出.

Table 2 Parameter estimation and comparison results of partial selected model (DS1)

表 2 部分选择的模型参数估计和比较结果(数据集 1)

模型	参数估计	52%故障数据,从 1 周~42 周		48%故障数据,从 43 周~81 周	
		MSE_{1fit}	KD_{fit}	$MSE_{1predict}$	$KD_{predict}$
Proposed new model (PM)	a=445.9429, b=0.000946, δ=0.2228, d=0.8579	90.1	0.057 6	336.5	0.072
G-O model with a single CP (GOMSC)	a=669.845, b_1=0.01848, b_2=0.01876, τ=31	104.9	0.083 7	818.3	0.112
Inflection S-shaped model with a single CP (ISSMSC)	a=424.529, b_1=0.06822, b_2=0.06749, β=2.16, τ=31	96.6	0.061 1	437.5	0.098
Goel-Okumoto (G-O)	a=727.566, b=0.01652	102.8	0.089 1	1 140.4	0.127
Delay S-shaped (DSS)	a=382.06, b=0.09341	250.1	0.089 4	1 627.3	0.160

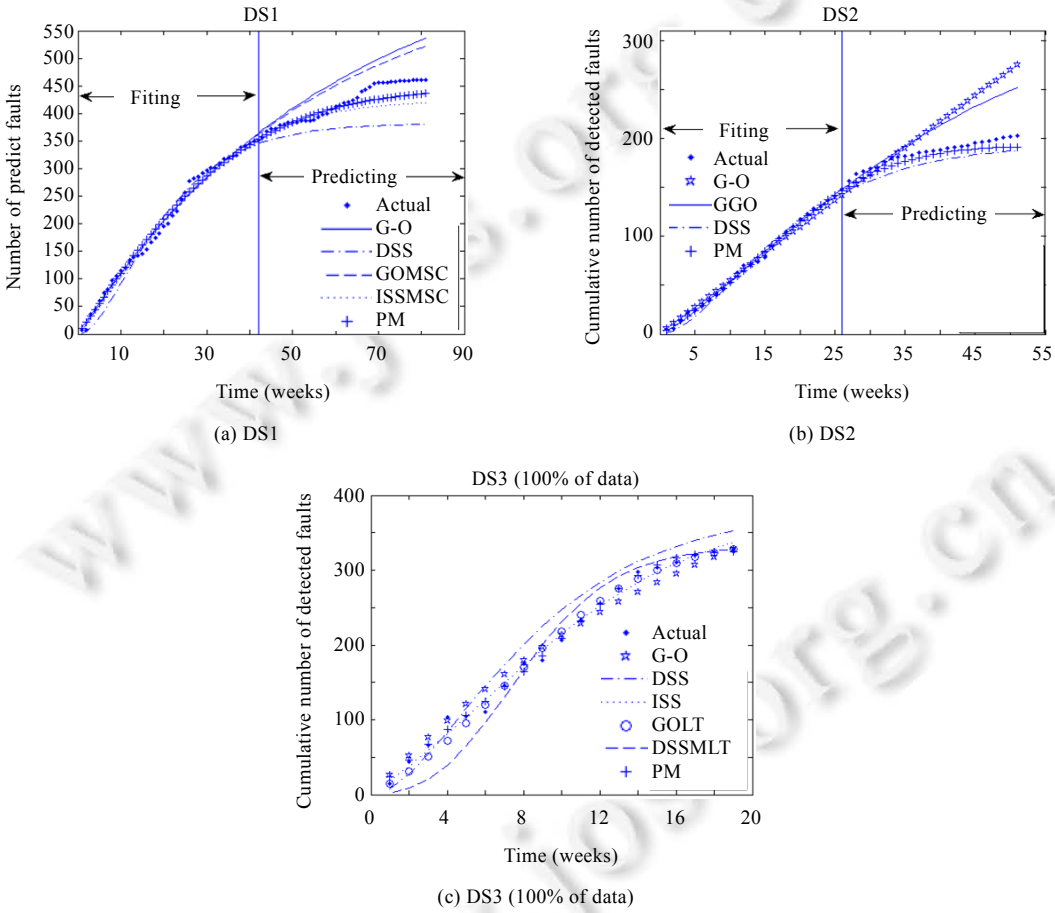

Fig.2 Estimated cumulative number of detected faults using DS1, DS2 and DS3 for partial selected models, respectively

图 2 用故障数据集 DS1,DS2 和 DS3 估计的部分选择的模型累计检测到的故障数量情况

图 3(a)给出了 PM 使用数据集 1 进行故障拟合和预测的 95%的置信区间.从图 3(a)可以清楚地看到:PM 准确地拟合了故障数量,并很好地预测了软件未来发生故障的失效行为.图 3(a)也显示出,实际观察到的故障数量很好地落在了提出模型的 95%的置信区间内.

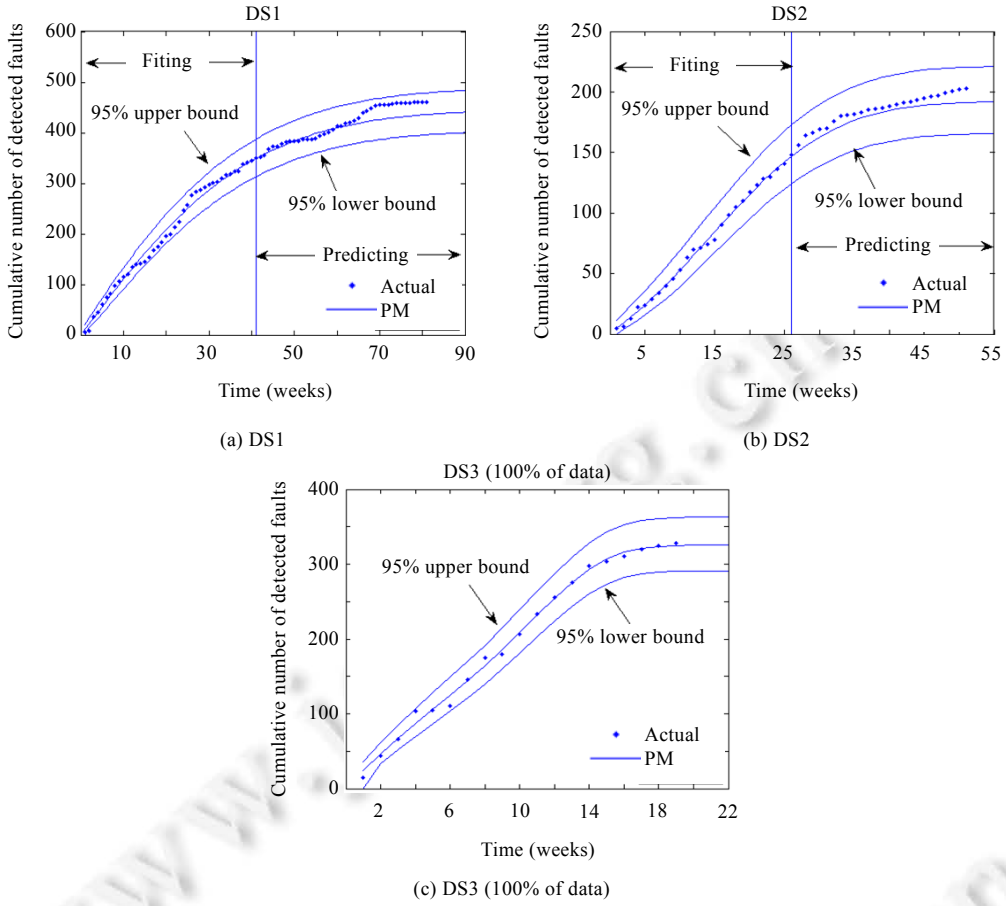

Fig.3 Plots of mean value functions and 95% s-confidence bounds for the actual data

图 3 用实际故障数据得出 PM 的均值函数和 95%的置信区间

2) 数据集 2(DS2)

相似地,从表 3 同样能够得出,PM 拟合及预测的 MSE_1 和 KD 值都小于其他模型,例如 G-O,GGO 和 DSS 模型,它们分别有 MSE_{1fit}=6.4,KD_{1fit}=0.04,$MSE_{1predict}$=53.4,$KD_{1predict}$=0.0399.

Table 3 Parameter estimation and comparison results of partial selected model (DS2)

表 3 部分选择的模型参数估计和比较结果(数据集 2)

模型	参数估计	50%故障数据,从 1 周~26 周		50%故障数据,从 27 周~51 周	
		MSE_{1fit}	KD_{fit}	$MSE_{1predict}$	$KD_{predict}$
Proposed new model (PM)	a=192.1848, b=0.000899, δ=0.2159, d=1.3621	6.4	0.04	53.4	0.039 9
Goel-Okumoto (G-O)	a=5039.9, b=0.0011	22.4	0.053 4	715.2	0.251 9
Generalized Goel model (GGO)	a=346.6926, b=0.0093, c=1.2555	6.6	0.043 3	403.9	0.179
Delay S-shaped (DSS)	a=194.5197, b=0.1011	17.8	0.079 5	88.9	0.079 9

与其他模型相比,MSE_1 和 KD 拟合值第二小的是 Generalized Goel model(GGO),它们的值分别是 6.6 和 0.043 3.MSE_1 和 KD 预测值第二小的是 Delay S-shaped(DSS),它们的值分别是 88.9 和 0.079 9.因此可以合理地得出,PM 有很好的拟合和预测性能.这些结果也可以从图 2(b)中看到.图 3(b)给出了 PM 使用数据集 2 进行故障拟合和预测的 95%置信区间.从图 3(b)可以看到:PM 很好地拟合了故障数量并预测了软件存在故障的数量.图 3(b)也显示出,PM 的 95%置信区间很好地包括了实际观察到的故障数量发生情况.

3) 数据集 3(DS3)

从表 4 可以得出 PM 的拟合能力情况.我们采用 100%的故障数据进行了相关模型的拟合性能评测,结果是:PM 的 *MSE* 和 *KD* 值都小于其他模型,包括 G-O,DSS,ISS,GOLT 和 DSSMLT 等模型.其中,我们提出的拟合 *MSE* 和 *KD* 值分别为 42.6 和 0.048 2;第二小的是 Inflection S-shaped(ISS),拟合 *MSE* 和 *KD* 值分别为 82.7 和 0.073 7;Delay S-shaped(DSS)有最大的拟合 *MSE* 和 *AIC* 值,分别为 640.8 和 0.11,这说明,PM 同样有很好的拟合故障数据集的能力.这些结果也可以从图 2(c)得到.图 3(c)显示了 PM 使用数据集 3 进行故障拟合的 95%置信区间,从图 3(c)可以清晰地看到:PM 准确地拟合了故障数量情况.图 3(c)也显示,实际发生的故障数量较好地落在了所提出模型的 95%置信区间内.

Table 4 Parameter estimation and comparison results of partial selected model (DS3)

表 4 部分选择的模型参数估计和比较结果(数据集 3)

模型	参数估计	100%故障数据	
		MSE_{1fit}	KD_{fit}
Proposed new model (PM)	a=325.5336, b=2.7261×10^{-5}, δ=0.39105, d=3.6438	42.6	0.048 2
Goel-Okumoto (G-O)	a=513.15, b=0.0537	222.1	0.095 2
Delay S-shaped (DSS)	a=384.05, b=0.219	640.8	0.112
Inflection S-shaped (ISS)	a=382.3631, b=0.1788, β=2.8865	82.7	0.073 7
G-O with logistic TEF (GOLT)	a=395.57, b=0.0416, N=54.84, A=13.03, α=0.226	114.1	0.094 4
Delay S-shaped model with Logistic TEF (DSSMLT)	a=339.96, b=0.121, N=54.84, A=13.03, α=0.226	634.8	0.103

总之,从以上实验可以合理地得出:与其他模型相比,PM 不论在故障拟合还是在故障预测能力上都比其他模型要好;而且 PM 还具有很强的鲁棒性,也就是说,对于不同的故障数据集都有很好的适用性.

3.4.2 全部模型拟合和预测性能比较

为了充分和有效地检验 PM 在实际软件测试中的拟合和预测性能,我们把故障数据集 4~数据集 8 分别划分为 4 类,分别是 50%,75%,90%和 100%;其中,故障数据集的 50%,75%和 90%是指用故障数据集前 50%、前 75%和前 90%的故障数据来估计模型的参数,并用故障数据集后 50%、后 25%和后 10%的故障数据来评估模型的预测能力.而 100%故障数据是指用全部的故障数据来验证模型的拟合故障数据能力.需要注意的是,我们用文献[34]中的方法确定了故障数据集 DS4~DS8 的移动点(CP),它们分别是:在 DS4 中,τ=20;在 DS5 中,τ=25;在 DS6 中, τ=70;在 DS7 中,τ=20;在 DS8 中,τ=70.

1) 数据集 4(DS4)

从表 5 可以看到:

- 在 50%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 值最小,其次是 Inflection S-shaped with a single CP(ISSMSC),最差的是 Inflection S-shaped(ISS).第二小的 $MSE_{1predict}$ 值(230.8)是 PM(86)的两倍还多.同样,PM 的 *AIC* 值也是最小的,仅为 141.3;
- 在 75%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 30.6 和 246.1.$MSE_{1predict}$ 值第二小的是 Generalized Goel model(GGO),为 50.8.*AIC* 值第二小的是 Inflection S-shaped (ISS),为 254.97.$MSE_{1predict}$ 和 *AIC* 值都最大的是 G-O model with a single CP(GOMSC),分别为 1 661.1 和 288.4;
- 在 90%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 61.8 和 301.3.$MSE_{1predict}$ 值第二小的是 G-O with logistic TEF(GOLT),为 71.3.*AIC* 值第二小的是 Inflection S-shaped with a single CP(ISSMSC),为 309.8.$MSE_{1predict}$ 和 *AIC* 值都最大的是 G-O model with a single CP (GOMSC),分别为 1 298.3 和 353.7.因此能够得出:PM 的预测能力好于其他模型,且拥有最低的 *AIC* 值;
- 在 100%的数据集中,与其他模型相比,PM 的 MSE_{1fit} 和 *AIC* 值都是最小的,分别是 35.7 和 335.2.MSE_{1fit} 值第二小的是 Delay S-shaped(DSS)和 Delay S-shaped model with a single CP(DSSMSC),它们都是 40.9.*AIC* 值第二小的是 Inflection S-shaped with a single CP(ISSMSC),是 343.8.MSE_{1fit} 值最大的是 G-O with logistic TEF(GOLT),为 437.1.*AIC* 值最大的是 Inflection S-shaped(ISS)和 G-O model with a single

CP(GOMSC),它们都是 378.8.这说明,PM 有很强的故障拟合能力.

Table 5 Comparison results of all selected model for DS4

表 5 全部选择的模型比较结果(数据集 4)

模型	50%的数据集		75%的数据集		90%的数据集		100%的数据集	
	$MSE_{\backslash predict}$	*AIC*	$MSE_{\backslash predict}$	*AIC*	$MSE_{\backslash predict}$	*AIC*	$MSE_{\backslash predict}$	*AIC*
Goel-Okumoto (G-O)	1 609.28	158.09	182.31	286.39	73.69	351.69	386.6	376.8
Generalized Goel model (GGO)	941.1	159.4	50.8	264	221.1	315.4	45.7	353.2
Delay S-shaped (DSS)	1541.5	158.66	82.94	263.37	235.88	314.46	40.9	353
Inflection S-shaped (ISS)	1 628.88	160.09	1 041	254.97	148.33	323.19	432.7	378.8
G-O with logistic TEF (GOLT)	1482	157.6	194.9	285	71.3	349.3	437.1	375.3
Delay S-shaped model with logistic TEF (DSSMLT)	1 585.2	158.7	109.2	263.3	240.3	314.4	40.9	353
G-O model with a single CP (GOMSC)	3 141.5	160.1	1661.1	288.4	1 298.3	353.7	325.7	378.8
Inflection S-shaped model with a single CP (ISSMSC)	230.8	158.3	668.9	257	342.3	309.8	104.4	343.8
Proposed new model (PM)	86	141.3	30.6	246.1	61.8	301.3	35.7	335.2

图 4(a)~图 4(d)分别给出了本文所提出模型使用数据集 4 的 50%,75%,90%和 100%进行故障拟合和预测的 95%的置信区间,从中也能看出,PM 很好地拟合和预测了实际的软件测试中软件故障发生的情况.从图 4 可以清晰地看到:从图 4(a)~图 4(d),PM 的 95%置信区间逐渐从一般到很好地包括了实际观察到的故障数量发生情况.

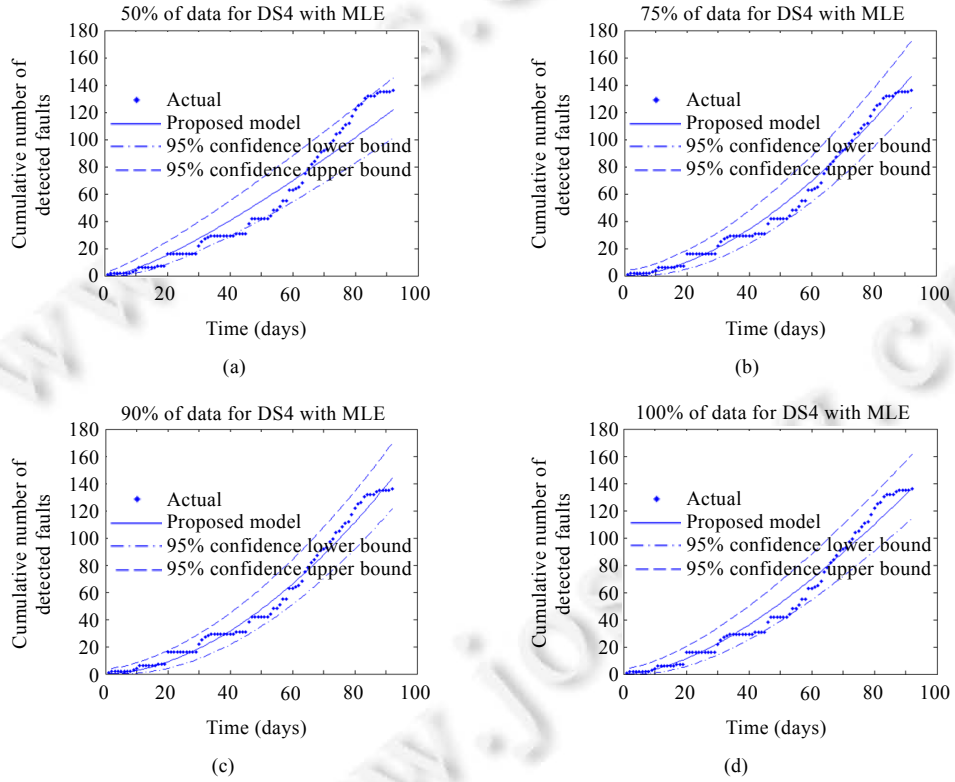

Fig.4 Plots of mean value functions and 95% s-confidence bounds for the actual data (DS4)

图 4 用实际故障数据(数据集 4)得出 PM 的均值函数和 95%的置信区间

2) 数据集 5(DS5)

相似地,从表 6 可以看到:

- 在 50%的数据集中,与其他模型相比,PM 的 $MSE_{\backslash predict}$ 值最小,是 31.5;其次是 Generalized Goel model

(GGO),为 37.2;最差的是 G-O with logistic TEF(GOLT),为 309.2.同样,PM 的 *AIC* 值也是最小的,仅为 184.6;

- 在 75%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 13.6 和 249.4. $MSE_{1predict}$ 值第二小的是 G-O with logistic TEF(GOLT),为 20.8.最差的是 Inflection S-shaped with a single CP(ISSMSC),是 875.2.*AIC* 值第二小的是 Generalized Goel model(GGO),为 259.3.*AIC* 值最大的是 G-O model with a single CP(GOMSC),为 277.2;
- 在 90%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 0.6 和 320.2. $MSE_{1predict}$ 值第二小的是 Delay S-shaped(DSS),为 0.81.*AIC* 值第二小的也是 Delay S-shaped (DSS),为 322.8.$MSE_{1predict}$ 最大的是 Inflection S-shaped with a single CP(ISSMSC),为 321.4.*AIC* 值最大的是 G-O model with a single CP(GOMSC),为 335.7.所以我们能够得出:PM 的预测能力好于其他模型,并且拥有最低的 *AIC* 值;
- 在 100%的数据集中,与其他模型相比,PM 的 MSE_{1fit} 和 *AIC* 值都是最小的,分别是 9.1 和 346.7.MSE_{1fit} 值第二小的是 Delay S-shaped(DSS),为 11.*AIC* 值第二小的也是 Delay S-shaped(DSS),为 350.8.MSE_{1fit} 和 *AIC* 值最大的是 G-O model with a single CP(GOMSC),分别是 477.9 和 363.8.这可以说明,PM 有很强的故障拟合能力.

图 5(a)~图 5(d)分别给出了 PM 使用数据集 5 的 50%,75%,90%和 100%进行故障拟合和预测的 95%的置信区间,也可看出,PM 很好地拟合和预测了实际的软件测试中软件故障发生的数量.图 5 中,从图 5(a)~图 5(d),实际观察到的故障数量较好地落在了 PM 的 95%上下界中.

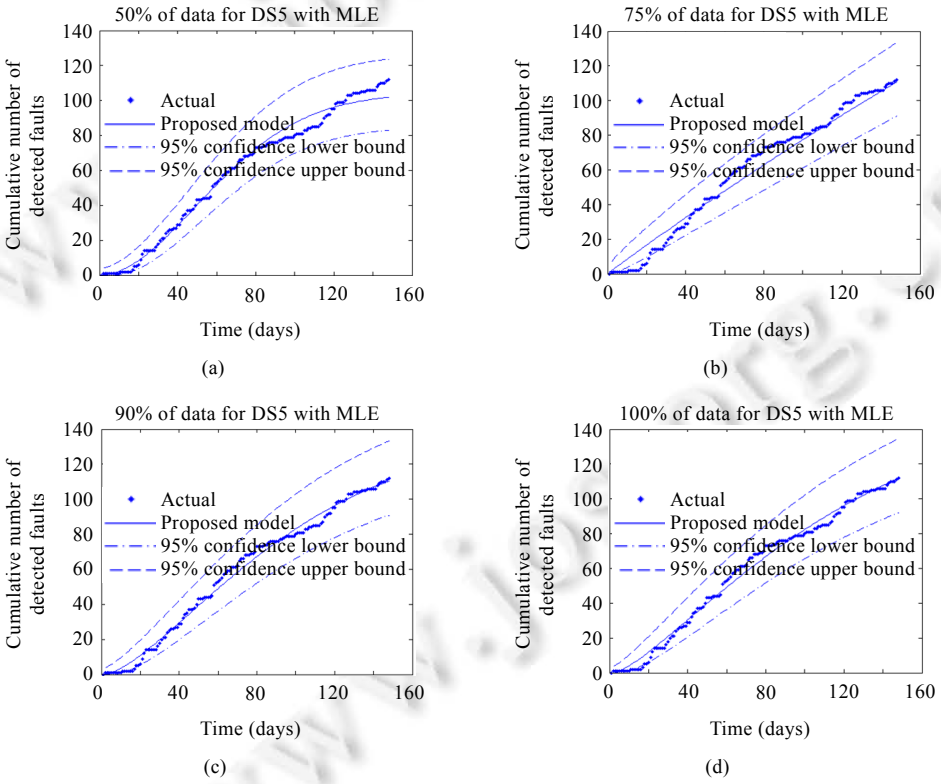

Fig.5 Plots of mean value functions and 95% s-confidence bounds for the actual data (DS5)

图 5 用实际故障数据(数据集 5)得出 PM 的均值函数和 95%的置信区间

Table 6 Comparison results of all selected model for DS5

表 6 全部选择的模型比较结果(数据集 5)

模型	50%的数据集		75%的数据集		90%的数据集		100%的数据集	
	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*
Goel-Okumoto (G-O)	215.98	202.8	19.8	275.2	7.6	333.7	25.7	361.8
Generalized Goel model (GGO)	37.2	192.5	236.3	259.3	1.1	325.9	12.5	353.9
Delay S-shaped (DSS)	275.3	190.8	120.7	259.8	0.81	322.8	11	350.8
Inflection S-shaped (ISS)	215.3	204.8	255.7	260.9	7.8	335.6	16.2	359.1
G-O with logistic TEF (GOLT)	309.2	201.9	20.8	275.1	7.1	333.6	25.5	361.7
Delay S-shaped model with logistic TEF (DSSMLT)	205.5	190.8	127.7	259.7	0.9	322.9	11.4	350.9
G-O model with a single CP (GOMSC)	150.3	204.8	681.2	277.2	190.9	335.7	477.9	363.8
Inflection S-shaped model with a single CP (ISSMSC)	293.9	195.7	875.2	262.9	321.4	332.8	283.8	361.1
Proposed new model (PM)	31.5	184.6	13.6	249.4	0.6	320.2	9.1	346.7

3) 数据集 6(DS6)

从表 7 可以看到:

- 在 50%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 值最小,为 609.2;其次是 G-O with logistic TEF (GOLT),是 644.6;最差的是 Delay S-shaped(DSS).同样,PM 的 *AIC* 值也是最小的,仅为 411.5;第二小的 *AIC* 值为 G-O with logistic TEF(GOLT),为 417.2;
- 在 75%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 14.9 和 694.6. $MSE_{1predict}$ 值第二小的是 Delay S-shaped(DSS),为 35.1.*AIC* 值第二小的是 Inflection S-shaped with a single CP(ISSMSC),为 696.5.$MSE_{1predict}$ 值最大的是 G-O model with a single CP(GOMSC),为 1 480.6. *AIC* 值最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 722.1;
- 在 90%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 2 和 852.4. $MSE_{1predict}$ 值第二小的是 Delay S-shaped(DSS),为 2.4.*AIC* 值第二小的是 Inflection S-shaped with a single CP(ISSMSC),为 861.4.$MSE_{1predict}$ 和 *AIC* 值都最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 184.7 和 885.4.因此我们能够得出:PM 与其他模型相比,预测能力均是最强的,并且拥有最低的 *AIC* 值;
- 在 100%的数据集中,与其他模型相比,PM 的 MSE_{1fit} 和 *AIC* 值都是最小的,分别是 26.6 和 945.6.MSE_{1fit} 值第二小的是 Delay S-shaped(DSS),为 34.3.*AIC* 值第二小的是 Inflection S-shaped with a single CP (ISSMSC),为 949.1.MSE_{1fit} 值和 *AIC* 值都最大的是 Delay S-shaped model with logistic TEF(DSSMLT), 分别是 244.7 和 973.1.这可以充分地说明,PM 与其他模型相比具有很强的故障拟合能力.

Table 7 Comparison results of all selected model for DS6

表 7 全部选择的模型比较结果(数据集 6)

模型	50%的数据集		75%的数据集		90%的数据集		100%的数据集	
	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*
Goel-Okumoto (G-O)	675	417.5	364.3	702.7	2.7	867.6	227.4	955.4
Generalized Goel model (GGO)	701.3	419.4	44.3	700.3	9	863.6	94.2	952.5
Delay S-shaped (DSS)	2 205.9	436.6	35.1	722	2.4	885.3	34.3	973
Inflection S-shaped (ISS)	711.5	419.4	57.6	704.7	17.3	869.6	120.4	957.4
G-O with logistic TEF (GOLT)	644.6	417.2	51.6	701.9	2.8	866.8	162.5	955
Delay S-shaped model with logistic TEF (DSSMLT)	2 175.1	436.7	16.9	722.1	184.7	885.4	244.7	973.1
G-O model with a single CP (GOMSC)	1536	419.4	1 480.6	704.7	3.3	869.6	244.5	957.4
Inflection S-shaped model with a single CP (ISSMSC)	1 567.5	420.2	136.7	696.5	99.7	861.4	170.8	949.1
Proposed new model (PM)	609.2	411.5	14.9	694.6	2	852.4	26.6	945.6

图 6(a)~图 6(d)分别给出了 PM 使用数据集 6 的 50%,75%,90%和 100%进行故障拟合和预测的 95%的置信区间,从中我们也能看出,PM 很好地拟合和预测了实际的软件测试中软件故障发生的行为.从图 6 可以清晰地

看到,PM 的 95%的上下界从图 6(a)~图 6(d)逐渐更好地包括了实际观察到的故障数量发生情况.

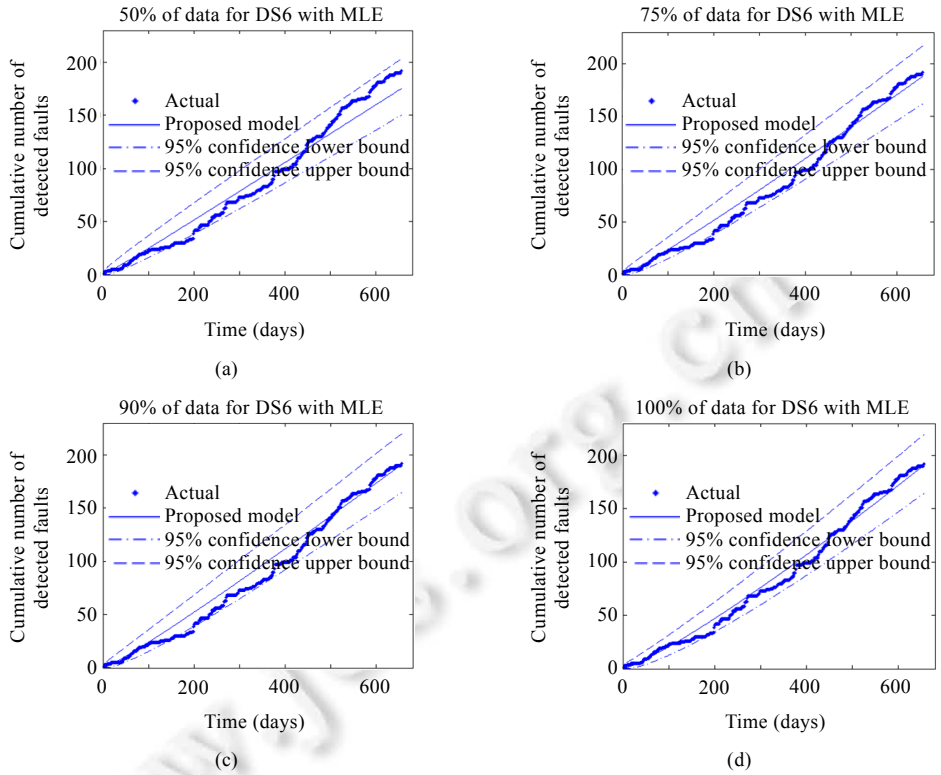

Fig.6 Plots of mean value functions and 95% s-confidence bounds for the actual data (DS6)

图 6 用实际故障数据(数据集 6)得出 PM 的均值函数和 95%的置信区间

4) 数据集 7(DS7)

从表 8 可以看到:

- 在 50%的数据集中,与其他模型相比,PM 具有最小的 $MSE_{1predict}$ 值和 *AIC* 值,分别为 51.4 和 511.2.第二小的 $MSE_{1predict}$ 值是 Inflection S-shaped(ISS),为 80.5.最大的 $MSE_{1predict}$ 值是 Delay S-shaped model with logistic TEF(DSSMLT),为 783.7.第二小的 *AIC* 值为 G-O with logistic TEF(GOLT),为 517.最大的 *AIC* 值是 Delay S-shaped model with logistic TEF(DSSMLT),为 531.1;
- 在 75%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 8.2 和 982.9. $MSE_{1predict}$ 值第二小的是 Generalized Goel model(GGO),为 9.3.*AIC* 值第二小的是 Inflection S-shaped (ISS),为 986.9.$MSE_{1predict}$ 值和 *AIC* 值都最大的是 Delay S-shaped model with logistic TEF (DSSMLT),分别是 470.5 和 1 062.4;
- 在 90%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 1.6 和 1 155.4. $MSE_{1predict}$ 值第二小的是 G-O with logistic TEF(GOLT),为 2.4.*AIC* 值第二小的是 Generalized Goel model(GGO),为 1 160.1.$MSE_{1predict}$ 值最大的是 Goel-Okumoto(G-O),为 659.5.*AIC* 值最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 1 250.1.因此我们可以合理地得出:PM 与其他模型相比,预测能力均是最强的,并且拥有最低的 *AIC* 值;
- 在 100%的数据集中,与其他模型相比,PM 的 MSE_{1fit} 和 *AIC* 值都是最小的,分别是 41.6 和 1 250.4. MSE_{1fit} 值第二小的是 Inflection S-shaped(ISS),为 55.8.*AIC* 值第二小的是 Generalized Goel model (GGO),为 1 251.2.MSE_{1fit} 值最大的是 G-O with logistic TEF(GOLT),为 404.7.这可以充分说明,PM 与其

他模型相比具有很强的故障拟合性能.

Table 8 Comparison results of all selected model for DS7

表 8 全部选择的模型比较结果(数据集 7)

模型	50%的数据集		75%的数据集		90%的数据集		100%的数据集	
	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*
Goel-Okumoto (G-O)	680.2	728	283.3	990.7	659.5	1 161.6	61.4	1 252.4
Generalized Goel model (GGO)	257.6	724.9	9.3	990.9	17.4	1 160.1	94.4	1 251.2
Delay S-shaped (DSS)	1212.5	807.1	446	1 061.8	11.4	1 249.6	135.6	1 345.8
Inflection S-shaped (ISS)	508.9	730	357.7	986.9	2.9	1 163.3	55.8	1 254.3
G-O with logistic TEF (GOLT)	8 190.9	728	38.8	990.5	2.4	1 161.6	404.7	1 252.4
Delay S-shaped model with logistic TEF (DSSMLT)	1 440.6	808.1	470.5	1 062.4	18.3	1 250.1	136.1	1 346.1
G-O model with a single CP (GOMSC)	287.5	730	413.8	992.7	3.8	1 163.6	76.6	1 254.4
Inflection S-shaped model with a single CP (ISSMSC)	282.6	732	438.5	988.9	30.9	1 165.3	102.4	1 256.3
Proposed new model (PM)	222.8	722.3	8.2	982.9	1.6	1 155.4	41.5	1 250.4

图 7(a)~图 7(d)分别给出了 PM 使用数据集 7 的 50%,75%,90%和 100%进行故障拟合和预测的 95%的置信区间,从中我们也能看出,PM 很好地拟合和预测了在实际的软件测试中软件故障发生的情况.图 7(a)~图 7(d)都显示出,PM 的 95%置信区间均较好地包括了实际观察到的故障数量发生情况.

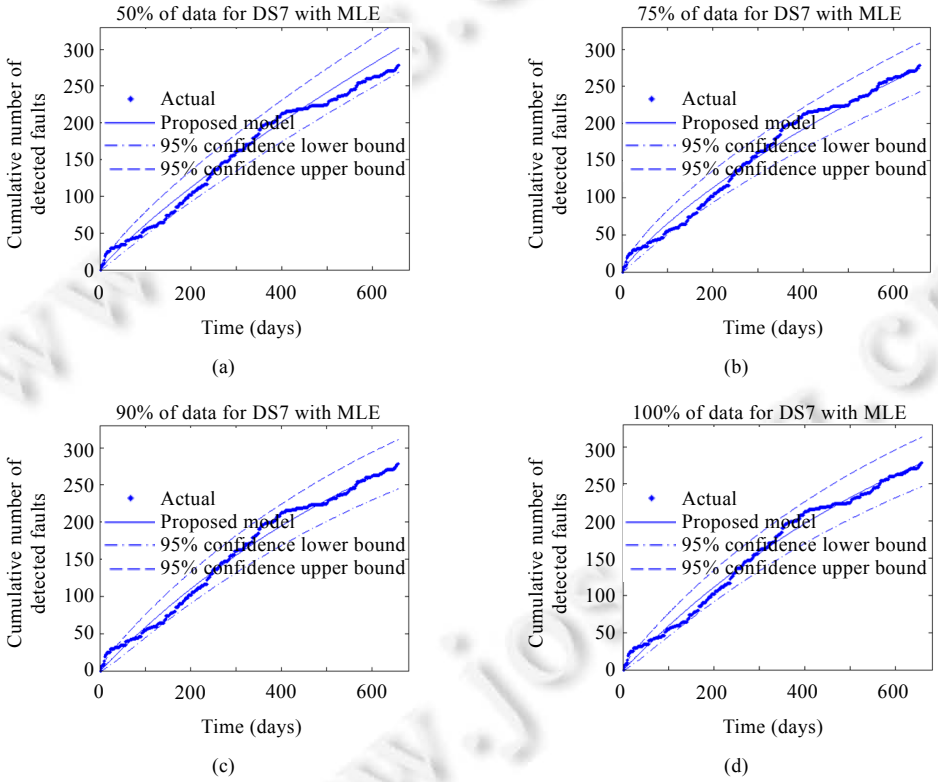

Fig.7 Plots of mean value functions and 95% s-confidence bounds for the actual data (DS7)

图 7 用实际故障数据(数据集 7)得出 PM 的均值函数和 95%的置信区间

5) 数据集 8(DS8)

相似地,从表 9 可以看到:

- 在 50%的数据集中,与其他模型相比,PM 具有最小的 $MSE_{1predict}$ 值和 *AIC* 值,分别为 51.4 和 511.2.第二

小的 $MSE_{1predict}$ 值是 Inflection S-shaped(ISS),为 80.5.最大的 $MSE_{1predict}$ 值是 Delay S-shaped model with logistic TEF(DSSMLT),为 783.7.第二小的 *AIC* 值是 G-O with logistic TEF(GOLT),为 517.最大的 *AIC* 值是 Delay S-shaped model with logistic TEF(DSSMLT),为 531.1;

- 在 75%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 24.6 和 742.1. $MSE_{1predict}$ 值第二小的是 Goel-Okumoto(G-O),为 25.9.*AIC* 值第二小的是 G-O with logistic TEF (GOLT),为 744.$MSE_{1predict}$ 值最大的是 G-O model with a single CP(GOMSC),为 710.7.*AIC* 值最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 760.6;
- 在 90%的数据集中,与其他模型相比,PM 的 $MSE_{1predict}$ 和 *AIC* 值都是最小的,分别是 7.4 和 901.6. $MSE_{1predict}$ 值第二小的是 Delay S-shaped(DSS),为 8.5.*AIC* 值第二小的是 Goel-Okumoto(G-O)和 G-O with logistic TEF(GOLT),它们都是 906.7.$MSE_{1predict}$ 值最大的是 G-O model with a single CP(GOMSC),为 370.*AIC* 值最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 932.7.因此我们可以合理地得出:PM 与其他模型相比有较好的预测能力,并且拥有最低的 *AIC* 值;
- 在 100%的数据集中,与其他模型相比,PM 的 MSE_{1fit} 和 *AIC* 值都是最小的,分别是 10.5 和 955.MSE_{1fit} 值第二小的是 Inflection S-shaped(ISS),为 11.9.*AIC* 值第二小的也是 Inflection S-shaped(ISS),为 962. MSE_{1fit} 值最大的是 Inflection S-shaped model with a single CP(ISSMSC),为 185.8.*AIC* 值最大的是 Delay S-shaped model with logistic TEF(DSSMLT),为 982.5.这可以充分地说明,PM 与其他模型相比具有很强的故障拟合性能.

图 8(a)~图 8(d)分别给出了本文所提出模型使用数据集 8 的 50%,75%,90%和 100%进行故障拟合和预测的 95%的置信区间,从中我们也能看出,PM 能够在实际的软件测试中很好地拟合和预测软件故障发生的情况.

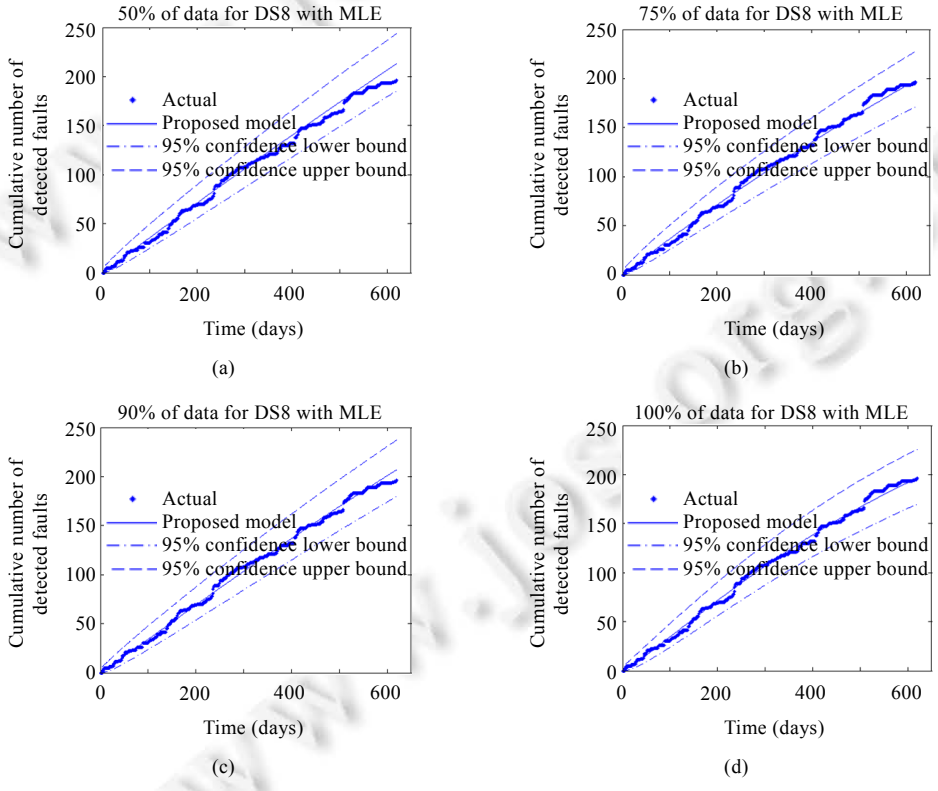

Fig.8 Plots of mean value functions and 95% s-confidence bounds for the actual data (DS8)

图 8 用实际故障数据(数据集 8)得出 PM 的均值函数和 95%的置信区间

图 8(a)~图 8(d)都显示出,PM 的 95%置信区间均很好地包括了实际观察到的故障数量发生情况.

Table 9 Comparison results of all selected model for DS8

表 9 全部选择的模型比较结果(数据集 8)

模型	50%的数据集		75%的数据集		90%的数据集		100%的数据集	
	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*	$MSE_{1predict}$	*AIC*
Goel-Okumoto (G-O)	171.8	517.1	25.9	744.1	54.2	906.7	18.8	962.2
Generalized Goel model (GGO)	202.4	518.9	52.9	745.4	56.6	908.6	12.5	963
Delay S-shaped (DSS)	727.2	530.8	256.6	760.4	8.5	932.2	48.8	982.2
Inflection S-shaped (ISS)	80.5	519.1	129.6	744.8	62.6	908.7	11.9	962
G-O with logistic TEF (GOLT)	236	517	26.3	744	57.1	906.7	18.5	962.2
Delay S-shaped model with logistic TEF (DSSMLT)	783.7	531.1	270.5	760.6	14	932.7	50.4	982.5
G-O model with a single CP (GOMSC)	245.9	519.1	710.7	746.1	370	908.7	91.1	964.2
Inflection S-shaped model with a single CP (ISSMSC)	435.9	520.4	85.2	746.8	79.8	910.7	185.8	964
Proposed new model (PM)	51.4	511.2	24.6	742.1	7.4	901.6	10.5	955

总之,从以上的实验可以合理地得出:虽然有些模型在测试的某些阶段和 PM 有相似的拟合和预测结果,但是总体来说,PM 在测试的整个过程中都有很好的拟合和预测性能;并且能够精准地拟合和预测故障数量,而且还能够很好地适应复杂的测试情况.因此,与其他模型相比,在实际的软件测试中,利用 PM 进行相应的软件可靠性评估更具有重要的指导意义.

4 结束语

在本文中,我们给出了一个考虑故障检测率不规则变化的 NHPP 软件可靠性增长模型.特别是,我们假设故障检测率在故障检测过程中随时间发生不规则变化,这种假设符合实际的故障测试过程中故障检测率受到各种因素的影响而发生不规则变化的客观规律.为了验证 PM 的性能,我们选用了不同软件可靠性模型、多种在不同条件下收集的故障数据集以及 3 种不同的模型评比标准,并进行了 PM 与其他模型的故障拟合和预测能力比较.实验结果表明:与其他模型相比,PM 有更好的故障拟合和预测能力;并且在实际软件测试中用来评估软件可靠性更具有重要的指导意义.

在未来的研究中,我们准备在完美调试的基础上进一步研究故障检测率在不完美调试中的不规则变化情况,以及由此来建立相应的故障检测率,为不规则变化的不完美软件调试模型.

致谢 在此,我们向对本文的工作给予支持和建议的老师和同行表示感谢.

References:

[1] Musa JD, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill, 1989. 32−241.

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

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

[4] Ohba M. Inflection S-shaped software reliability growth models. In: Osaki S, Hatoyama Y, eds. Proc. of the Stochastic Models in Reliability Theory. Berlin: Springer-Verlag, 1984. 144−162. [doi: 10.1007/978-3-642-45587-2_10]

[5] Misra PN. Software reliability analysis. IBM Systems Journal, 1983,22:262−270. [doi: 10.1147/sj.223.0262]

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

[7] 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]

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

[9] Wang WL, Pan D, Chen MH. Architecture-Based software reliability modeling. Journal of System Software, 2006,79(1):132−146. [doi: 10.1016/j.jss.2005.09.004]

[10] Hu QP, Xie M, Ng SH, Levitin G. Robust recurrent neural network modeling for software fault detection and correction prediction. Reliability Engineering & System Safety, 2007,92(3):332−340. [doi: 10.1016/j.ress.2006.04.007]

[11] Jeske DR, Zhang X. Some successful approaches to software reliability modeling in industry. Journal of System Software, 2005, 74(1):85−99. [doi: 10.1016/j.jss.2003.10.024]

[12] Goel AL. Software reliability models: Assumptions, limitations and applicability. IEEE Trans. on Software Engineering, 1985,SE-11(12):1411−1423. [doi: 10.1109/TSE.1985.232177]

[13] Amman PE, Brilliant SS, Knight J. The effect of imperfect error detection on reliability assessment via life testing. IEEE Trans. on Software Engineering, 1994,20(2):142−148. [doi: 10.1109/32.265635]

[14] 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]

[15] Xie M, Yang B. A study of the effect of imperfect debugging on software development cost. IEEE Trans. on Software Engineering, 2003,29(5):471−473. [doi: 10.1109/TSE.2003.1199075]

[16] 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):2241−2252. [doi: 10.1109/TSMCA.2003.812597]

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

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

[19] 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(4):365−377. [doi: 10.1142/S0218539304001579]

[20] 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]

[21] Zhao M. Change-Point problems in software and hardware reliability. Communication in Statistics-Theory and Methods, 1993, 22(3):757−768. [doi: 10.1080/03610929308831053]

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

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

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

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

[26] Zhao J, Liu HW, Cui G, Yang XZ. Software reliability growth model with change-point and environmental function. Journal of System Software, 2006,79(11):1578−1587. [doi: 10.1016/j.jss.2006.02.030]

[27] 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]

[28] Wong E. Stochastic Processes in Information and Dynamical Systems. New York: McGraw-Hill, 1971.

[29] Arnold L. Stochastic Differential Equations—Theory and Applications. New York: John Wiley & Sons, 1974.

[30] 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]

[31] Xiao X, Dohi T. Wavelet shrinkage estimation for non-homogeneous Poisson process based software reliability models. IEEE Trans. on Reliability, 2013,62(1):211−225. [doi: 10.1109/TR.2013.2240897]

[32] Wang J, Wu Z, Shu Y. Analysis of the debugging model based on probabilistic state transaction. Journal of Software, 2013,8(11): 2697−2705. [doi: 10.4304/jsw.8.11.2697-2705]

[33] Musa JD. Software Reliability Engineering: More Reliable Software, Faster and Cheaper. 2nd ed., Bloomington: Author-House, 2004.

[34] Xie M. Software reliability models—Past, present and future. In: Proc. of the Recent Advances in Reliability Theory: Methodology, Practice and Inference. Birkhauser, 2000. 323−340. [doi: 10.1007/978-1-4612-1384-0_21]

[35] Tamura Y, Yamada S. Optimization analysis for reliability assessment based on stochastic differential equation modeling for open source software. Int'l Journal of Systems Science, 2009,40(4):429−438. [doi: 10.1080/00207720802556245]

[36] Kanoun K, Martini M, Souza J. A method for software reliability analysis and prediction application to the TROPICO-R switching system. IEEE Trans. on Software Engineering, 1991,17(4):334−344. [doi: 10.1109/32.90433]

[37] Jeske DR, Zhang X, Pham L. Adjusting software failure rates that are estimated from test data. IEEE Trans. on Reliability, 2005, 54(1):107−114. [doi: 10.1109/TR.2004.842531]

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

[39] Musa JD. Software reliability data. Report, Rome: Data and Analysis Center for Software, Rome Air Development Center, 1980.

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

[41] Akaike H. A new look at statistical model identification. IEEE Trans. on Automatic Control, 1974,AC-19(6):716−723. [doi: 10.1109/TAC.1974.1100705]

王金勇(1974－),男,黑龙江鸡西人,博士生,主要研究领域为容错计算,软件可靠性.

舒燕君(1981－),女,博士,讲师,CCF 会员,主要研究领域为容错计算,软件可靠性.

吴智博(1954－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为容错计算,移动计算.

张展(1978－),男,博士,副教授,CCF 会员,主要研究领域为容错计算,移动计算.