# 考虑用户行为和排错延迟的软件运行可靠性增长模型

杨剑锋[1] 赵 明[2] 胡文生[3]

(贵州理工学院大数据学院 贵阳 550003)[1]

(耶夫勒大学可持续发展技术学院 耶夫勒 80176)[2] (贵州理工学院信息网络中心 贵阳 550003)[3]

**摘 要** 传统的软件可靠性模型大多都假设软件测试环境和运行环境相同,也就是使用软件测试阶段的失效数据来预测软件运行可靠性。众所周知,软件固有故障的排除能提高系统可靠性,然而另一种现象就是随着用户对软件熟悉程度的提高,软件的失效率也会降低。文中研究了软件固有故障检测过程、固有故障纠正过程和外在失效过程的特征,建立了考虑用户行为和排错延迟下的软件运行可靠性增长模型。通过一组来自于开源软件用户缺陷跟踪系统中的真实数据进行数值分析,实验结果表明提出的模型具有较好的效果。

**关键词** 软件运行可靠性,用户行为,排错延迟,固有故障,外在故障

**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.01.033

## Software Operational Reliability Growth Model Considering End-user Behavior and Fault-correction Time Delay

YANG Jian-feng[1] ZHAO Ming[2] HU Wen-sheng[3]

(School of Data Science, Guizhou Institute of Technology, Guiyang 550003, China)[1]

(Faculty of Technology and Sustainable Development, University of Gävle, Gävle 80176, Sweden)[2]

(Center of Information & Network Technology, Guizhou Institute of Technology, Guiyang 550003, China)[3]

**Abstract** Most of traditional software reliability models assume that the testing environment and the opera-ting environment are the same, that is, the software reliability model using failure data during the testing phase can predict the operational reliability. It is well known that correcting bugs will improve software reliability, while another phenomenon occurs: the failure rate has decreased as the users are more familiar with the system. In this paper, the inherent fault-detection process (IFDP), inherent fault-correction process (IFCP) and external fault-detection process (EFDP) were discussed. Moreover, a software operational reliability growth model considering end-user behavior and fault-correction time delay was proposed. By using the real data from end-users bug tracking data for open source software, the numerical results show that the proposed model is useful and powerful.

**Keywords** Software operational reliability, User behavior, Fault-correction time delay, Inherent fault, External fault

## 1 引言

软件公司为了开发出高质量、高安全、高满意度的产品,往往花费开发成本的50%~75%对软件进行测试,以排除故障,提高软件的可靠性[1]。利用测试阶段的失效数据建立的软件可靠性模型没有考虑到测试环境与运行环境的差别,而这种环境差异会影响软件运行可靠性的评估与预测。

考虑到软件使用阶段和测试阶段环境的不同,一些学者利用环境因子调整测试阶段和使用阶段环境的差别,以更好地区分测试可靠性和运行可靠性。文献[2]给出了测试可靠性与运行可靠性的概念,对测试可靠性和运行可靠性进行了量化分析比较,然后分析了这两种可靠性对软件最优发布时间的影响。在软件测试阶段,由于加速测试的影响,用软件可靠性增长模型预测所得到的运行阶段软件失效强度很难真实地反映软件在实际运行阶段的失效强度[3]。因此,一些学者引入了环境因子(Environmental Factor)的概念,利用环境因子降低测试环境与运行环境的差别给软件可靠性估计精度造成的误差。文献[3-6]定义环境因子为测试阶段的故障检测率与运行阶段的故障检测率之比,并假设环境因子是不随时间变化的一个随机变量,建立了测试阶段和运行阶段的可靠性预测模型。而文献[7]对环境因子的推导方法进行了改进,并假设环境因子与时间有关,建立了有环境因子的变点可靠

到稿日期:2018-02-01 返修日期:2018-04-20 本文受贵州省科学技术基金计划(黔科合J字[2015]2064号,黔科合基础[2016]1066号,黔科合LH字[2016]7108号)基金,高层次人才科研启动经费项目(XJGC20150106)资助。

**杨剑锋**(1986-),男,博士,副教授,CCF会员,主要研究方向为软件可靠性、应用统计,E-mail:jfyang1@163.com(通信作者);**赵 明**(1962-),男,博士,教授,主要研究方向为可靠性工程、应用统计;**胡文生**(1969-),男,博士,副教授,主要研究方向为软件可靠性与可维护性。

性预测模型。以上利用环境因子来建立可靠性模型的方法取得了一些研究成果。

另一部分学者利用软件发布后的软件销售量和失效数据来预测软件的运行可靠性。这类研究考虑了软件测试可靠性与运行可靠性的区别，并认为只有软件固有故障的排除能提高软件可靠性，但是并未考虑终端用户使用软件的行为习惯。实际研究表明，软件一旦投入市场，它的可靠度不仅受软件缺陷本身的影响，还会受用户对软件的熟练程度以及环境配置的影响[8-9]。对此，Jalote 等[10]于 2004 年通过对某软件产品发布后每月销售量及失效数的记录和观测，提出了一种软件发布后的软件可靠性预测模型。2008 年，Jalote 等[11]再次对原有模型进行了修改，得到的改进模型在预测能力上有所改善。软件版本的升级会影响软件的可靠性，一些学者提出了多版本的软件可靠性模型[12-14]、不完美排错和排错时间延迟的二维多版本软件可靠性模型[15-16]以及考虑变点的可靠性模型[17]。一些学者研究了故障排错时间延迟的多版本开源软件可靠性模型[18]，以及模型的一般性框架[19]。

另外，文献[20]考虑到软件的用户量对软件可靠性增长的影响，利用创新扩散模型估计软件的用户量，建立软件运行可靠性模型。文献[21]建立了基于随机森林的软件运行可靠性模型。文献[22]利用统计分析方法研究不同环境因子对软件开发与可靠性的影响。一些学者在软件运行环境的不确定性情形下，提出了一种三参数故障检测软件可靠性模型[23]，以及基于软件测试覆盖的故障检测率可靠性模型[24]。

相对于测试可靠性评估与预测方法来讲，对软件发布后的运行可靠性进行评估与预测的方法较少，其主要有两个方面的原因：1)软件发布后用户的使用数据很难再获得，缺少模型分析的数据；2)软件发布后的各种环境要比测试环境复杂得多，使得对运行可靠性的预测不准确。目前，随着软件工程技术的快速发展，已有不少大型软件开发公司提供了用户错误报告系统，如微软的 Windows 错误报告系统(WER)、Mozilla 的 Bugzilla 缺陷跟踪管理系统等。错误报告系统中的缺陷数据、Web 网络报文和服务日志包含有非结构化或者半结构化的复杂数据，因此如何利用这些数据对软件的运行可靠性进行动态评估与预测是当前研究的热点问题。

## 2　考虑用户行为和排错延迟的软件运行可靠性增长模型

### 2.1　模型的假设

本文在已有可靠性模型的基础上，创新性地考虑了用户的行为对系统可靠性的影响。在研究中考虑了软件运行中的固有失效和外在失效两种失效类型，即导致软件系统失效的原因可能是软件残留的固有故障被激发，或是外在的用户行为。在固有失效和外在失效的基础上，为了建立非齐次泊松过程类软件运行可靠性增长模型，本文先做如下基本假设：

1)软件运行过程中的固有失效和外在失效两种失效过程是统计独立的。

2)软件发布后，软件开发商会给出新的补丁，用户会及时安装和使用。补丁的作用是剔除已造成软件失效的固有故障，打补丁不会引进新的错误。

3)由于打补丁的行为受软件开发人员等因素影响，因此固有故障的排除需要一定的时间。外在故障是受用户熟悉程度的影响，其故障会立即得到排除，或者相对于整个时间周期来讲故障排除的时间可以忽略不计。

4)故障纠正过程是故障检测过程的一个时间延迟过程。固有失效和外在失效过程都可以用非齐次泊松过程来描述。

### 2.2　模型的建立

#### 2.2.1　可靠性模型框架

记 $m_{1d}(t)$ 为固有故障检测过程的均值函数，$m_{1c}(t)$ 为固有故障纠正过程的均值函数，$m_2(t)$ 为外在失效过程的均值函数。软件系统的可靠性增长体现在软件固有故障的纠正和用户对软件的逐渐熟悉(外在失效的排除)上。因此，可以得到软件系统故障累积的均值函数为：

$$m(t)=m_{1c}(t)+m_2(t) \tag{1}$$

软件系统的失效强度函数 $\lambda(t)$ 为：

$$\lambda(t)=m'(t)=m'_{1c}(t)+m_2'(t)=\lambda_{1c}(t)+\lambda_2(t) \tag{2}$$

其中，$\lambda_{1c}(t)$ 为固有故障纠正过程的失效强度函数，$\lambda_2(t)$ 为外在失效过程的失效强度函数。软件系统的可靠性表现为在当前时间 $t$ 下，在将来时间区间 $(t,t+\Delta t)$ 内不发生失效的概率。根据非齐次泊松过程的性质，可以得到系统的可靠性函数为：

$$R(\Delta t|t)=\exp\{-[m(t+\Delta t)-m(t)]\} \tag{3}$$

G-O 模型是一种使用得最为广泛的 NHPP 类可靠性增长模型[25]，其均值函数 $m(t)$ 和强度函数 $\lambda(t)$ 分别为：

$$\begin{aligned} m(t)&=a[1-\exp(-rt)]\\ \lambda(t)&=ar\exp(-rt)\end{aligned} \tag{4}$$

其中，$a$ 为软件系统潜伏的故障总数，$r$ 为故障检测率。对于固有软件失效过程，本文需要考虑故障排错延迟给软件可靠性带来的影响，可以认为故障纠正过程是故障检测过程的一个时间延迟过程，记随机变量 $\Delta t$ 为故障纠正时间。由于 $\Delta t$ 具有随机性，故障随机延迟过程的失效强度函数 $\lambda_{1c}^*$ 和均值函数 $m_{1c}^*$ 均为随机变量，分别为：

$$\lambda_{1c}^*=\begin{cases}\lambda_{1d}(t-\Delta t), & \Delta t\leqslant t\\ 0, & \Delta t>t\end{cases} \tag{5}$$

$$m_{1c}^*=\begin{cases}m_{1d}(t-\Delta t), & \Delta t\leqslant t\\ 0, & \Delta t>t\end{cases} \tag{6}$$

为了获得 FCP 的失效强度函数和均值函数，文献[26]对随机变量 $\lambda_{1c}^*$ 取期望，得到失效强度函数 $\lambda_{1c}(t)$ 的表达式，然后根据均值函数和失效强度函数的关系可以求出均值函数 $m_{1c}(t)$ 的表达式：

$$\lambda_{1c}(t)=E[\lambda_{1c}^*]=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x \tag{7}$$

$$m_{1c}(t)=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot f(x)\mathrm{d}x\mathrm{d}\tau \tag{8}$$

文献[26]中的方法首先对延迟过程的失效强度函数 $\lambda_{1c}^{*}$ 取期望,求得 FCP 过程的强度函数 $\lambda_{1c}(t)$,然后根据均值函数和强度函数之间的关系求得 FCP 过程的均值函数 $m_{1c}(t)$。类似地,可以对随机变量 $m_{1c}^{*}$ 取期望,得到 FCP 模型的均值函数 $m_{1c}(t)$,然后计算 FCP 过程的强度函数 $\lambda_{1c}(t)$。为了说明这两种方法是等价的,首先引入以下定理。

**定理 1** $\lambda_{1c}(t)=E[\lambda_{1c}^{*}]$ 当且仅当 $m_{1c}(t)=E[m_{1c}^{*}]$。

证明:1)充分性

$$\begin{aligned}m_{1c}(t)&=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot f(x)\mathrm{d}x\mathrm{d}\tau\\&=\int_0^t\int_x^t\lambda_{1d}(\tau-x)\cdot f(x)\mathrm{d}\tau\mathrm{d}x\\&=\int_0^t m_{1d}(t-x)\cdot f(x)\mathrm{d}x=E[m_{1c}^{*}]\end{aligned}$$

2)必要性

因为:

$$m_{1c}(t)=E[m_{1c}^{*}]=\int_0^t m_{1d}(t-x)\cdot f(x)\mathrm{d}x$$

所以有:

$$\begin{aligned}\lambda_{1c}(t)&=m'_{1c}(t)\\&=\int_0^t m'_{1d}(t-x)\cdot f(x)\mathrm{d}x+m_{1d}(t-t)\cdot f(t)\\&=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x=E[\lambda_{1c}^{*}]\end{aligned}$$

综上,有 $\lambda_{1c}(t)=E[\lambda_{1c}^{*}]\Leftrightarrow m_{1c}(t)=E[m_{1c}^{*}]$。

通过定理 1 可以看出,两种求 FCP 模型均值函数的结果一样,下文将不再对这两种方法进行区分。

本文首先考虑 $\Delta t$ 为指数随机变量和 Gamma 随机变量这两种情形,其他随机变量可以进行类似的推广。

### 2.2.2 指数分布可靠性模型

1987 年,Musa 通过实际案例发现软件在测试阶段的故障纠正时间近似服从指数分布[27]。因此,本文首先考虑软件的故障纠正时间服从指数分布的情形,即有:

$$\Delta t\sim\exp(\mu)\tag{9}$$

指数分布的概率密度函数为:

$$f(x)=\mu\mathrm{e}^{-\mu x},\mu>0,x>0\tag{10}$$

在给定故障检测过程的失效强度函数 $\lambda_{1d}(t)$ 后,可以得到 FCP 的失效强度函数 $\lambda_{1c}(t)$ 为:

$$\begin{aligned}\lambda_{1c}(t)&=E[\lambda_{1c}^{*}]=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x\\&=\int_0^t\lambda_{1d}(t-x)\cdot\mu\mathrm{e}^{-\mu x}\mathrm{d}x\end{aligned}\tag{11}$$

进一步,可得 FCP 的均值函数 $m_{1c}(t)$ 为:

$$m_{1c}(t)=E[m_{1c}^{*}]=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot\mu\mathrm{e}^{-\mu x}\mathrm{d}x\mathrm{d}\tau\tag{12}$$

根据式(1)—式(2)、式(11)—式(12)可得系统的强度函数和均值函数分别为:

$$\lambda(t)=\lambda_{1c}(t)+\lambda_2(t)=\int_0^t\lambda_{1d}(t-x)\cdot\mu\mathrm{e}^{-\mu x}\mathrm{d}x+\lambda_2(t)\tag{13}$$

$$m(t)=m_{1c}(t)+m_2(t)=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot\mu\mathrm{e}^{-\mu x}\mathrm{d}x\mathrm{d}\tau+m_2(t)\tag{14}$$

特别地,可以得到 G-O 模型下 FCP 的失效强度函数为:

$$\lambda_{1c}(t)=\int_0^t a_1r_1\mathrm{e}^{-r_1(t-x)}\cdot\mu\mathrm{e}^{-\mu x}\mathrm{d}x=\begin{cases}a_1r_1^2t\mathrm{e}^{-r_1t}, & \mu=r_1\\\dfrac{a_1r_1\mu}{\mu-r_1}(\mathrm{e}^{-r_1t}-\mathrm{e}^{-\mu t}), & \mu\neq r_1\end{cases}\tag{15}$$

G-O 模型下 FCP 的均值函数为:

$$m_{1c}(t)=\begin{cases}a_1[1-(1+r_1t)\mathrm{e}^{-r_1t}], & \mu=r_1\\a_1[1-\dfrac{\mu}{\mu-r_1}\mathrm{e}^{-r_1t}+\dfrac{r_1}{\mu-r_1}\mathrm{e}^{-\mu t}], & \mu\neq r_1\end{cases}\tag{16}$$

G-O 模型下系统的强度函数和均值函数分别为:

$$\lambda(t)=\begin{cases}a_1r_1^2t\mathrm{e}^{-r_1t}+a_2r_2\mathrm{e}^{-r_2t}, & \mu=r_1\\\dfrac{a_1r_1\mu}{\mu-r_1}(\mathrm{e}^{-r_1t}-\mathrm{e}^{-\mu t})+a_2r_2\mathrm{e}^{-r_2t}, & \mu\neq r_1\end{cases}\tag{17}$$

$$m(t)=\begin{cases}a_1[1-(1+r_1t)\mathrm{e}^{-r_1t}]+a_2(1-\mathrm{e}^{-r_2t}), & \mu=r_1\\a_1[1-\dfrac{\mu}{\mu-r_1}\mathrm{e}^{-r_1t}+\dfrac{r_1}{\mu-r_1}\mathrm{e}^{-\mu t}]+a_2(1-\mathrm{e}^{-r_2t}), & \mu\neq r_1\end{cases}\tag{18}$$

### 2.2.3 Gamma 分布可靠性模型

为了给出更精确的 FCP 模型,需要更广泛的故障纠正时间分布。Gamma 分布是一个最好的选择,它是指数分布的推广,可以解释为故障纠错由很多步骤组成,即:

$$\Delta t\sim\Gamma(\alpha,\beta)\tag{19}$$

Gamma 分布的概率密度函数为:

$$f(x)=\frac{1}{\beta^{\alpha}\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta},\alpha,\beta>0\tag{20}$$

其中,$\Gamma(\alpha)=\int_0^\infty t^{\alpha-1}\mathrm{e}^{-t}\mathrm{d}t$。

与指数分布的故障纠正时间类似,给定故障检测过程的失效强度函数 $\lambda_{1d}(t)$ 后,可以得到 FCP 的失效强度函数 $\lambda_{1c}(t)$ 为:

$$\begin{aligned}\lambda_{1c}(t)&=E[\lambda_{1c}^{*}]=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x\\&=\int_0^t\lambda_{1d}(t-x)\cdot\frac{1}{\beta^{\alpha}\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x\end{aligned}\tag{21}$$

进一步可得 FCP 的均值函数 $m_{1c}(t)$ 为:

$$\begin{aligned}m_{1c}(t)&=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=E[m_{1c}^{*}]\\&=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot\frac{1}{\beta^{\alpha}\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x\mathrm{d}\tau\end{aligned}\tag{22}$$

根据式(1)和式(2)、式(21)—式(22)可得系统的强度函数和均值函数分别为:

$$\begin{aligned}\lambda(t)&=\lambda_{1c}(t)+\lambda_2(t)\\&=\int_0^t\lambda_{1d}(t-x)\cdot\frac{1}{\beta^{\alpha}\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x+\lambda_2(t)\end{aligned}\tag{23}$$

$$m(t)=m_{1c}(t)+m_2(t)$$

$$=\int_0^t\int_0^\tau \lambda_{1d}(\tau-x)\cdot\frac{1}{\beta^\alpha\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x\mathrm{d}\tau+m_2(t) \tag{24}$$

特别地，可以得到 G-O 模型下 FCP 的失效强度函数 $\lambda_{1c}(t)$ 为：

$$\begin{aligned}\lambda_{1c}(t)&=\int_0^t a_1r_1\mathrm{e}^{-r_1(t-x)}\cdot\frac{1}{\beta^\alpha\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x\\&=\frac{a_1r_1}{\beta^\alpha\Gamma(\alpha)}\cdot\mathrm{e}^{-r_1t}\cdot\int_0^t x^{\alpha-1}\mathrm{e}^{(r_1-1/\beta)x}\mathrm{d}x\\&=\frac{a_1r_1}{(1-\beta r_1)^\alpha}\cdot\mathrm{e}^{-r_1t}\cdot\gamma(\alpha,\frac{1-\beta r_1}{\beta}t)\end{aligned} \tag{25}$$

其中，$\gamma(s,x)=\frac{1}{\Gamma(s)}\int_0^x t^{s-1}\mathrm{e}^{-t}\mathrm{d}t$。

可得 G-O 模型下 FCP 的均值函数为：

$$\begin{aligned}m_{1c}(t)&=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=E[m_{1c}^*]\\&=\int_0^t a_1(1-\mathrm{e}^{-r_1(t-x)})\cdot\frac{1}{\beta^\alpha\cdot\Gamma(\alpha)}x^{\alpha-1}\mathrm{e}^{-x/\beta}\mathrm{d}x\\&=a_1[\gamma(\alpha,\frac{t}{\beta})-\frac{\mathrm{e}^{-r_1t}}{(1-\beta r_1)^\alpha}\cdot\gamma(\alpha,\frac{1-\beta r_1}{\beta}t)]\end{aligned} \tag{26}$$

G-O 模型下系统的强度函数和均值函数分别为：

$$\lambda(t)=\frac{a_1r_1}{(1-\beta r_1)^\alpha}\cdot\mathrm{e}^{-r_1t}\cdot\gamma(\alpha,\frac{1-\beta r_1}{\beta}t)+a_2r_2\mathrm{e}^{-r_2t} \tag{27}$$

$$m(t)=a_2(1-\mathrm{e}^{-r_2t})+a_1[\gamma(\alpha,\frac{t}{\beta})-\frac{\mathrm{e}^{-r_1t}}{(1-\beta r_1)^\alpha}\cdot\gamma(\alpha,\frac{1-\beta r_1}{\beta}t)] \tag{28}$$

#### 2.2.4　考虑用户行为和一般排错延迟的可靠性模型框架

给定 FDP 过程的失效强度函数 $\lambda_{1d}(t)$ 或 $m_{1d}(t)$ 后，可以得到 FCP 过程的失效强度函数 $\lambda_{1c}(t)$ 为：

$$\lambda_{1c}(t)=E[\lambda_{1c}^*]=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x \tag{29}$$

其中，$f(x)$ 为故障纠正时间随机变量 $\Delta t$ 的概率密度函数。

进一步可得 FCP 的均值函数 $m_{1c}(t)$ 为：

$$m_{1c}(t)=E[m_{1c}^*]=\int_0^t\lambda_{1c}(\tau)\mathrm{d}\tau=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot f(x)\mathrm{d}x\mathrm{d}\tau \tag{30}$$

根据式(1)－式(2)、式(29)－式(30)可得系统的强度函数和均值函数分别为：

$$\lambda(t)=\lambda_{1c}(t)+\lambda_2(t)=\int_0^t\lambda_{1d}(t-x)\cdot f(x)\mathrm{d}x+\lambda_2(t) \tag{31}$$

$$m(t)=m_{1c}(t)+m_2(t)=\int_0^t\int_0^\tau\lambda_{1d}(\tau-x)\cdot f(x)\mathrm{d}x\mathrm{d}\tau+m_2(t) \tag{32}$$

其中，$\lambda_2(t)$ 和 $m_2(t)$ 分别为外在失效过程的强度函数和均值函数。根据可靠性模型的一般性框架，类似地可以得到 Weibull 分布等排错随机延迟下的可靠性模型，本文不再一一描述。

## 3　参数估计与模型评估准则

由模型假设可知，固有失效过程和外在失效过程是统计独立的，因此可以分成两部分来对模型的参数进行估计。固有失效过程包括 FDP 和 FCP 两个过程，通常采用最小二乘估计方法；外在失效过程可以用极大似然对参数进行估计。

### 3.1　固有失效过程中参数的最小二乘估计

对于 FDP 模型和 FCP 模型中的参数，常规的方法是采用最小二乘方法进行参数估计，即式(33)取最小值时参数所取的估计值。

$$\begin{aligned}S(\theta)&=\sum_{i=1}^k[(m_{1d}(t_i)-d_i)^2+(m_{1c}(t_i)-c_i)^2]\\&=\sum_{i=1}^k(m_{1d}(t_i)-d_i)^2+\sum_{i=1}^n(m_{1c}(t_i)-c_i)^2\end{aligned} \tag{33}$$

其中，$d_i$ 和 $c_i$ 分别为到时间 $t_i$ 为止所检测到的累计故障数和纠正的累计故障数，$t_i(i=1,2,\cdots,k)$ 为软件发布后的时间观测点，$\theta$ 为模型中的参数向量。

### 3.2　外在失效过程中参数的极大似然估计

极大似然估计法已被广泛应用到 NHPP 类软件可靠性增长模型中。外在失效过程模型的似然函数为：

$$L(\theta)=\prod_{i=1}^k\frac{[m_2(t_i)-m_2(t_{i-1})]^{(m_i-m_{i-1})}\cdot\exp\{-[m_2(t_i)-m_2(t_{i-1})]\}}{(m_i-m_{i-1})!} \tag{34}$$

其中，$m_i$ 为到时间 $t_i$ 为止检测到的累计外在失效数，$t_i(i=1,2,\cdots,k)$ 为软件发布后的时间观测点，$\theta$ 为模型中的参数向量，且 $m_0=t_0=0$。

进一步可得对数似然函数为：

$$\log L=\sum_{i=1}^k(m_i-m_{i-1})\ln[m_2(t_i)-m_2(t_{i-1})]-\sum_{i=1}^k[m_2(t_i)-m_2(t_{i-1})]-\sum_{i=1}^k\ln[(m_i-m_{i-1})!] \tag{35}$$

特别地，G-O 模型下的对数似然函数为：

$$\log L=\sum_{i=1}^k(m_i-m_{i-1})\cdot\ln[a_2(\mathrm{e}^{-r_2t_{i-1}}-\mathrm{e}^{-r_2t_i})]-[a_2(1-\mathrm{e}^{-r_2t_k})]-\sum_{i=1}^k\ln[(m_i-m_{i-1})!] \tag{36}$$

通常可以通过数值算法求得参数的极大似然估计。

### 3.3　模型评估准则

均方误差(MSE)已经被广泛应用于模型的拟合优度检验中。由于本文关系到 3 个失效过程，因此模型的 MSE 定义为：

$$\begin{aligned}MSE&=\frac{1}{3}[MSE_{1d}+MSE_{1c}+MSE_2]\\&=\frac{1}{3}\{\frac{1}{k}\sum_{i=1}^k[(m_{1d}(t_i)-d_i)^2]+\frac{1}{k}\sum_{i=1}^k[(m_{1c}(t_i)-c_i)^2]+\frac{1}{k}\sum_{i=1}^k[(m_2(t_i)-m_i)^2]\}\\&=\frac{1}{3k}\sum_{i=1}^k(m_{1d}(t_i)-d_i)^2+(m_{1c}(t_i)-c_i)^2+(m_2(t_i)-m_i)^2]\end{aligned} \tag{37}$$

## 4　案例分析

### 4.1　缺陷管理系统及实验数据

本文所需的故障数据来源于用户缺陷跟踪系统(Bug

Tracking System),它是一种错误报告系统,其中 Mozilla Bugzilla[1] 是一款用于软件缺陷追踪管理的网络应用程序,是当前比较流行的缺陷跟踪系统,由 Mozilla 基金会计划开发和应用。1998 年,网景公司开放其源代码,后以 Mozilla 公共许可证协议授权。Bugzilla 由 Perl 语言编写,能在多种数据库上运行,包括 MySQL 和 Oracle 等数据库。有许多组织和企业使用该产品(特别是开源软件),例如 Linux 内核开发团队、Apache 开发团队、GNOME 开发团队等。Bugzilla 缺陷跟踪管理系统能够帮助企业减少停机时间,提高开发效率,增强顾客的满意度,加强开发人员与用户之间的交流与沟通,并减少企业生产成本等。

Bugzilla 缺陷跟踪管理系统主要包括的数据字段有:ID,Comp,Status,Resolution,Opened,Last Resolved,Sev,OS,CC Count,Comments 等。字段中的报告分类状态(Status)和报告处理意见(Resolution)关系到失效数据的分类,而其他字段可以得到 Bug 所在的位置、发生和解决 Bug 的时间、一些简单的用户使用环境以及用户对某个 Bug 的关注度等。因此,根据不同的错误状态和错误处理报告可以将所有的 Bug 进行分类,从分类后的数据中提取软件失效数据,为建立可靠性模型提供数据基础。

根据 Bug 的不同状态和处理意见,可以将失效数据分成以下 8 种:1)"USERS"表示外在失效数据,在 Bugzilla 错误跟踪系统中表示错误状态为"WONTFIX"和"DUPLICATE"时所对应的失效,因为这些失效数据是由于用户使用不当或者对软件不熟悉导致的;2)"FDP"表示软件固有故障被检测到的数据,错误状态为"FIXED";3)"FCP"表示软件固有故障被纠正的数据,错误状态为"FIXED";4)"WORK"表示在当前环境下无法重现的错误,这可能与用户当前的各种环境配置有关,错误状态为"WORKSFORME";5)"INCO"表示当前描述的信息不完整,无法获得相应的错误,其可能与用户使用方法不正确或者环境配置有关,错误状态为"INCOMPLETE";6)"INVA"表示当前所描述的问题不是本软件的 Bug,也不是可靠性分析所需要的数据,错误状态为"INVALID";7)"UNCO"表示当前所描述的问题需要进一步确认,错误状态为"UNCONFIRMED";8)"ASS_NEW_REOP"表示当前已经确认,但是还没有解决的错误,错误状态为"ASSIGNED""NEW"或"REOPEN"。

Mozilla Firefox(火狐),是一个自由及开源的网页浏览器,使用 Gecko 引擎,由 Mozilla 基金会与社区共同开发。本文所需的故障数据来源于 Firefox 的用户缺陷跟踪系统[2]。根据上文介绍的失效数据分类方法,提取了开源软件 Firefox 的失效数据,共收集了软件释放后的 127 周失效数据,如表 1 所列,其中"UP"代表外在失效过程,"FDP"代表固有故障检测过程,"FCP"代表固有故障纠正过程。

表 1 Firefox 的 3 种不同故障数据

Table 1 Three different kinds of failure data of Firefox software

| 时间/周 | UP | FDP | FCP | 时间/周 | UP | FDP | FCP |
|---|---|---|---|---|---|---|---|
| 1 | 53 | 9 | 3 | 65 | 6 | 3 | 2 |
| 2 | 35 | 3 | 0 | 66 | 8 | 2 | 0 |
| 3 | 17 | 4 | 1 | 67 | 3 | 1 | 1 |
| 4 | 10 | 9 | 3 | 68 | 4 | 0 | 1 |
| 5 | 9 | 2 | 2 | 69 | 6 | 1 | 3 |
| 6 | 15 | 2 | 3 | 70 | 1 | 0 | 0 |
| 7 | 7 | 0 | 0 | 71 | 8 | 1 | 1 |
| 8 | 13 | 3 | 1 | 72 | 6 | 1 | 1 |
| 9 | 3 | 2 | 2 | 73 | 5 | 0 | 0 |
| 10 | 7 | 1 | 2 | 74 | 4 | 1 | 3 |
| 11 | 7 | 1 | 1 | 75 | 5 | 3 | 2 |
| 12 | 9 | 0 | 1 | 76 | 7 | 1 | 3 |
| 13 | 13 | 3 | 2 | 77 | 9 | 0 | 1 |
| 14 | 4 | 0 | 1 | 78 | 4 | 2 | 1 |
| 15 | 21 | 1 | 0 | 79 | 1 | 0 | 0 |
| 16 | 13 | 0 | 0 | 80 | 0 | 1 | 0 |
| 17 | 4 | 0 | 1 | 81 | 8 | 1 | 0 |
| 18 | 9 | 1 | 1 | 82 | 22 | 1 | 0 |
| 19 | 3 | 1 | 2 | 83 | 15 | 1 | 1 |
| 20 | 6 | 1 | 0 | 84 | 5 | 1 | 0 |
| 21 | 14 | 0 | 0 | 85 | 6 | 0 | 0 |
| 22 | 8 | 1 | 0 | 86 | 5 | 0 | 0 |
| 23 | 5 | 1 | 0 | 87 | 7 | 2 | 0 |
| 24 | 7 | 0 | 0 | 88 | 3 | 1 | 1 |
| 25 | 6 | 1 | 0 | 89 | 7 | 0 | 0 |
| 26 | 4 | 1 | 1 | 90 | 8 | 0 | 0 |
| 27 | 10 | 0 | 0 | 91 | 5 | 5 | 1 |
| 28 | 3 | 2 | 1 | 92 | 7 | 1 | 1 |
| 29 | 5 | 1 | 0 | 93 | 8 | 4 | 2 |
| 30 | 10 | 0 | 1 | 94 | 6 | 0 | 0 |
| 31 | 9 | 0 | 4 | 95 | 4 | 0 | 1 |
| 32 | 8 | 0 | 0 | 96 | 4 | 1 | 0 |
| 33 | 9 | 1 | 1 | 97 | 6 | 2 | 1 |
| 34 | 6 | 1 | 0 | 98 | 3 | 0 | 2 |
| 35 | 9 | 1 | 1 | 99 | 3 | 0 | 0 |
| 36 | 7 | 1 | 2 | 100 | 5 | 0 | 1 |
| 37 | 6 | 1 | 0 | 101 | 2 | 0 | 0 |
| 38 | 7 | 0 | 0 | 102 | 7 | 0 | 0 |
| 39 | 6 | 0 | 0 | 103 | 3 | 0 | 1 |
| 40 | 5 | 0 | 0 | 104 | 10 | 0 | 0 |
| 41 | 8 | 1 | 0 | 105 | 9 | 0 | 0 |
| 42 | 4 | 3 | 0 | 106 | 5 | 0 | 0 |
| 43 | 2 | 1 | 1 | 107 | 6 | 0 | 0 |
| 44 | 7 | 0 | 2 | 108 | 4 | 0 | 0 |
| 45 | 8 | 0 | 2 | 109 | 8 | 2 | 0 |
| 46 | 5 | 1 | 0 | 110 | 6 | 0 | 0 |
| 47 | 3 | 1 | 1 | 111 | 7 | 1 | 0 |
| 48 | 4 | 0 | 2 | 112 | 4 | 0 | 0 |
| 49 | 2 | 0 | 0 | 113 | 6 | 2 | 2 |
| 50 | 5 | 0 | 1 | 114 | 2 | 0 | 0 |
| 51 | 6 | 0 | 0 | 115 | 12 | 0 | 0 |
| 52 | 8 | 2 | 1 | 116 | 2 | 0 | 1 |
| 53 | 5 | 1 | 1 | 117 | 3 | 1 | 0 |
| 54 | 45 | 7 | 3 | 118 | 8 | 2 | 1 |
| 55 | 20 | 3 | 0 | 119 | 2 | 0 | 1 |
| 56 | 35 | 5 | 2 | 120 | 4 | 0 | 0 |
| 57 | 16 | 2 | 1 | 121 | 5 | 0 | 0 |
| 58 | 11 | 4 | 1 | 122 | 4 | 2 | 1 |
| 59 | 10 | 1 | 2 | 123 | 4 | 0 | 0 |
| 60 | 7 | 4 | 2 | 124 | 2 | 0 | 0 |
| 61 | 16 | 2 | 1 | 125 | 3 | 0 | 0 |
| 62 | 10 | 0 | 0 | 126 | 3 | 0 | 1 |
| 63 | 8 | 0 | 1 | 127 | 3 | 0 | 1 |
| 64 | 17 | 5 | 4 | | | | |

[1] http://www.bugzilla.org/

[2] http://bugzilla.mozilla.org/

### 4.2　模型性能对比分析

通过上文建立的模型以及参数估计方法，使用 4.1 节的真实数据可以得到如表 2 所列的参数估计结果及模型的均方误差。从表 2 可以看出，3 个模型的外在失效过程都是同一个模型，且 3 个模型的 *MSE* 相等，因此外在失效过程可靠性模型并不影响模型的整体性能。下面主要讨论固有失效过程的模型拟合效果，从表中可以看出，指数时间延迟模型的 $MSE_{1d}$(72.9716)，$MSE_{1c}$(82.0026)和 *MSE*(373.5272)最大，因此该模型的拟合效果最差；另外，Weibull 时间延迟模型的 $MSE_{1d}$(17.0387)比 Gamma 时间延迟模型的 $MSE_{1d}$(78.6452)小，而其 $MSE_{1c}$ 更大，两个模型各有拟合优势，即 Weibull 时间延迟模型对故障检测过程有更好的拟合效果，Gamma 时间延迟模型对故障纠正过程有更好的拟合效果；然而，从可靠性模型的整体拟合效果来看，Weibull 时间延迟模型的综合 *MSE*(348.4768)比 Gamma 时间延迟模型的综合 *MSE*(360.3741)小，这说明 Weibull 时间延迟模型的整体拟合误差较小。因此，从模型的整体性能来看，Weibull 时间延迟模型比指数时间延迟模型和 Gamma 时间延迟模型的效果好，指数时间延迟模型的拟合效果最差。

表 2　参数估计结果及均方误差

Table 2　Estimated parameters and *MSE*

| 模型 | 估计值（外在失效过程） | 估计值（固有失效过程） | *MSE* |
|---|---|---|---|
| 指数时间延迟模型 | $a_2=1528.41$<br>$r_2=0.0085$ | $a_1=189.8734$<br>$r_1=0.0110$<br>$\mu=0.0321$ | $MSE_{1d}=72.9716$<br>$MSE_{1c}=82.0026$<br>$MSE_2=965.6073$<br>$MSE=373.5272$ |
| Gamma 时间延迟模型 | | $a_1=180.8680$<br>$r_1=0.0121$<br>$\alpha=0.4417$<br>$\beta=81.4915$ | $MSE_{1d}=78.6452$<br>$MSE_{1c}=36.8697$<br>$MSE_2=965.6073$<br>$MSE=360.3741$ |
| Weibull 时间延迟模型 | | $a_1=218.3440$<br>$r_1=0.0090$<br>$\alpha=0.6476$<br>$\beta=0.1667$ | $MSE_{1d}=17.0387$<br>$MSE_{1c}=62.7843$<br>$MSE_2=965.6073$<br>$MSE=348.4768$ |

图 1 给出了基于 G-O 模型的外在失效过程的累计失效拟合图。图 2 和图 3 分别给出了指数时间延迟、Gamma 时间延迟和 Weibull 时间延迟模型下故障检测过程和故障纠正过程的累计失效拟合图。

相对于图 2 所示的故障检测过程来说，图 3 所示的故障纠正过程在指数时间延迟、Gamma 时间延迟和 Weibull 时间延迟下模型的拟合度相差较大，这说明故障纠正延迟时间的概率分布对可靠性的增长影响较大。

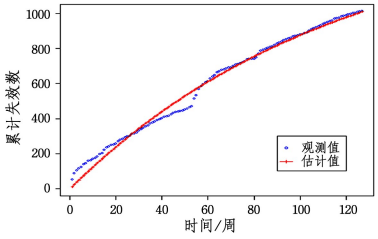


图 1　外在失效过程的模型拟合图

Fig. 1　Fitting plot of external fault-detection process

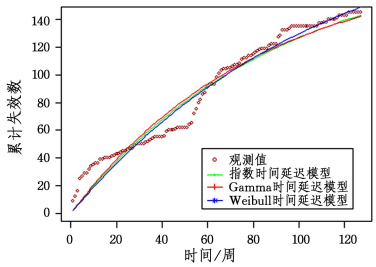


图 2　故障检测过程的模型拟合图

Fig. 2　Fitting plot of inherent fault-detection process

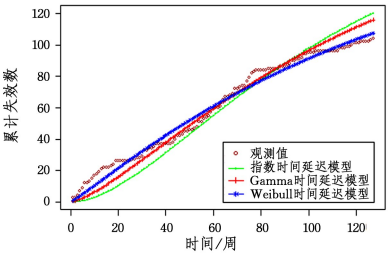


图 3　故障纠正过程的模型拟合图

Fig. 3　Fitting plot of inherent fault-correction process

**结束语**　本文分析了用户错误报告系统中的各种失效数据，根据软件测试环境和运行环境的差别，将软件释放后的失效分为外在失效和固有失效。外在失效是与用户行为相关的，例如用户对软件的熟悉程度、用户对软件的基本配置管理等；而固有失效是软件本身特有的故障，需要软件开发人员进行修改才能排除故障。在软件固有故障排除的过程当中，由于软件已经发布，软件开发商只有通过打补丁或者升级的方式对故障进行纠正，因此当软件的故障被检测到时，应当不会立即被排除，而在一段时间后通过打补丁或者升级的方式来排除故障。在此基础上，本文建立了考虑用户行为和排错延迟的软件运行可靠性增长模型。实验结果表明了所提模型的有效性，并且验证了 Weibull 时间延迟模型比指数时间延迟模型和 Gamma 时间延迟模型更好。

## 参 考 文 献

[1] O'CONNOR P D T, KLEYNER A. Practical reliability engineering[M]. John Wiley & Sons, 2012.

[2] YANG B, XIE M. A study of operational and testing reliability in software reliability analysis[J]. Reliability Engineering & System Safety, 2000, 70(3): 323-329.

[3] ZHANG X, JESKE D R, PHAM H. Calibrating software reliability models when the test environment does not match the user environment[J]. Applied Stochastic Models in Business & Industry, 2002, 18(1): 87-99.

[4] TENG X, PHAM H. A New Methodology for Predicting Software Reliability in the Random Field Environments[J]. IEEE Transactions on Reliability, 2006, 55(3): 458-468.

[5] PERSONA A. Systemability function to optimisation reliability in random environment[J]. International Journal of Mathematics in Operational Research, 2009, 1(3): 397-417.

[6] PHAM H. A new software reliability model with Vtub-shaped

fault-detection rate and the uncertainty of operating environments[J]. Optimization, 2014, 63(10): 1481-1490.

[7] ZHAO J, LIU H W, CUI G, et al. Software reliability growth model with change-point and environmental function[J]. Journal of Systems & Software, 2006, 79(11): 1578-1587.

[8] CHILLAREGE R, BIYANI S, ROSENTHAL J. Measurement of Failure Rate in Widely Distributed Software[C]// Proceedings of the International Symposium on Fault-Tolerant Computing. 1995.

[9] KAN S, MANLOVE D, GINTOWT B. Measuring system availability-field performance and in-process metrics[C]// Proceedings of 14th IEEE International Symposium on Software Reliability Engineering (ISSRE). 2003: 189-199.

[10] JALOTE P, MURPHY B. Reliability Growth in Software Products[C]// Proceedings of the International Symposium on Software Reliability Engineering. 2004: 47-53.

[11] JALOTE P, MURPHY B, SHARMA V S. Post-release reliability growth in software products[J]. ACM Transactions on Software Engineering and Methodology, 2008, 17(4): 1-20.

[12] KAPUR P K, AGGARWAL A G, NIJHAWAN N. A discrete SRGM for multi release software system[J]. International Journal of Industrial and Systems Engineering, 2014, 16(2): 143-155.

[13] KAPUR P K, PHAM H, AGGARWAL A G, et al. Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning[J]. IEEE Transactions on Reliability, 2012, 61(3): 758-768.

[14] KAPUR P K, PHAM H, ANAND S, et al. A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation[J]. IEEE Transactions on Reliability, 2011, 60(1): 331-340.

[15] KUMAR V, SAHNI R, SHRIVASTAVA A K. Two-dimensional multi-release software modelling with testing effort, time and two types of imperfect debugging[J]. International Journal of Reliability and Safety, 2016, 10(4): 368-388.

[16] KUMAR V, MATHUR P, SAHNI R, et al. Two-Dimensional Multi-Release Software Reliability Modeling for Fault Detection and Fault Correction Processes[J]. International Journal of Reliability Quality and Safety Engineering, 2016, 23(3): 1640002.

[17] TICKOO A, VERMA A K, KHATRI S K, et al. Modeling Two-Dimensional Framework for Multi-Upgradations of a Software with Change Point[J]. International Journal of Reliability Quality and Safety Engineering, 2016, 23(6): 570-574.

[18] YANG J, LIU Y, XIE M, et al. Modeling and analysis of reliability of multi-release open source software incorporating both fault detection and correction processes[J]. Journal of Systems and Software, 2016, 115: 102-110.

[19] LIU Y, XIE M, YANG J, et al. A New Framework and Application of Software Reliability Estimation Based on Fault Detection and Correction Processes[C]// Proceedings of the IEEE International Conference on Software Quality, Reliability and Security. 2015: 65-74.

[20] SHATNAWI O. Measuring Commercial Software Operational Reliability: An Interdisciplinary Modelling Approach[J]. Eksploat Niezawodn, 2014, 16(4): 585-594.

[21] ARAI Y, KIMURA M. Operational software reliability prediction by random forest based on development project data with qualitative variables[C]// Proceedings of the 21st ISSAT International Conference on Reliability and Quality in Design. 2015: 109-113.

[22] ZHU M, ZHANG X, HOANG P. A comparison analysis of environmental factors affecting software reliability[J]. Journal of Systems and Software, 2015, 109(1): 150-160.

[23] 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.

[24] LI Q, PHAM H. NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage[J]. Appl Math Model, 2017, 51(1): 68-85.

[25] GOEL A L, OKUMOTO K. Time-dependent error-detection rate model for software reliability and other performance measures[J]. IEEE Transactions on Reliability, 1979, 28(3): 206-211.

[26] XIE M, HU Q P, WU Y P, et al. A study of the modeling and analysis of software fault-detection and fault-correction processes[J]. Quality & Reliability Engineering International, 2007, 23(4): 459-470.

[27] MUSA J D, IANNINO A, OKUMOTO K. Software reliability-measurement, prediction, application[M]. McGraw-Hill Book, 1987.