Advances in Applied Mathematics 应用数学进展, 2022, 11(9), 6401-6410
Published Online September 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.119677

Hans汉斯

可叠加的随机微分方程软件可靠性模型

贺孟兰[1]，杨剑锋[1,2*]，丁　铭[1]

[1]贵州大学数学与统计学院，贵州　贵阳
[2]贵州理工学院大数据学院，贵州　贵阳

收稿日期：2022年8月13日；录用日期：2022年9月8日；发布日期：2022年9月15日

摘　要

随着软件系统规模的不断扩大，大多数复杂软件系统都由多个组件构成，组件的可靠性影响软件系统的可靠性。在实际中，软件故障跟踪系统中故障的报告存在随机性、不规律性以及各种不确定因素，因此故障检测过程可以看作是一个随机过程。本文应用伊藤型随机微分方程建立软件可靠性模型，利用各组件的故障数据来建立叠加的随机微分方程可靠性模型，对模型参数进行估计。最后，将两组实际软件项目的数据集应用于所提出的叠加随机微分方程可靠性模型，结果表明所提出的叠加随机微分方程可靠性模型有更优的效果。

关键词

软件可靠性模型，随机微分方程，叠加模型

Additive Stochastic Differential Equations Software Reliability Models

Menglan He[1], Jianfeng Yang[1,2*], Ming Ding[1]

[1]School of Mathematics and Statistics, Guizhou University, Guiyang Guizhou
[2]School of Data Science, Guizhou Institute of Technology, Guiyang Guizhou

Received: Aug. 13th, 2022; accepted: Sep. 8th, 2022; published: Sep. 15th, 2022

Abstract

With the increasing scale of software systems, most complex software systems are composed of multiple components, and the reliability of the components affects the reliability of the software system. In practice, there are randomness, irregularity and various uncertainties in the reporting

*通讯作者。

文章引用: 贺孟兰, 杨剑锋, 丁铭. 可叠加的随机微分方程软件可靠性模型[J]. 应用数学进展, 2022, 11(9): 6401-6410.
DOI: 10.12677/aam.2022.119677

of faults in software fault tracking systems, so the fault detection process can be regarded as a stochastic process. In this paper, the Itô type stochastic differential equation (SDE) is applied to build a software reliability model, and the fault data of each component is used to build a superimposed stochastic differential equation reliability model and to estimate the model parameters. Finally, data sets from two real software projects are applied to the proposed superposed stochastic differential equation reliability model, and the results show that the proposed superposed stochastic differential equation reliability model has better results.

Keywords

Software Reliability Model, Stochastic Differential Equation (SDE), Additive Model

1. 引言

近年来，计算机迅速发展，已经深入到人们生活的各个方面，大到国家系统小到日常生活。计算机软件也在不断的发展更新中，一旦计算机软件发生故障，就会对人们的生活和工作造成影响甚至可能造成巨大损失。例如 2021 年 11 月 24 日，阿里云在 Web 服务器软件阿帕奇下发现重大漏洞，该漏洞可能导致设备远程受控，引发敏感信息窃取、设备服务中断等严重危害，属于高危漏洞。这些由软件失效造成的影响不胜枚举，告诉我们软件可靠的重要性。

当前最常用的一类模型是基于 NHPP 的软件可靠性模型，该模型的特点是假设累计故障数服从非齐次泊松过程。原始的 NHPP 类模型是 G-O 模型，其后相继提出了 Delayed S-shaped、Inflection S-shaped 以及 Yamada 等经典模型，为后续的研究提供重要的参考。Wang 等[1]考虑故障检测与修正过程的时间相关性，结合了故障检测和修正过程建立软件可靠性模型，进行了数值实例研究；Kapur 等[2]考虑了测试团队在测试过程中不断获得的测试工作和知识等因素，提出了基于测试工作量的软件可靠性模型；Jain 等[3]引入多变点，在考虑不完全调试、测试工作函数和故障减少因子的基础上建立软件可靠性增长模型。以上 NHPP 模型都是在经典模型的基础上，结合不完美排错、测试工作量以及变点等进行研究。后续研究学者引入随机微分方程来建立软件可靠性模型。Tamura 等[4]考虑开源软件的活跃状态，提出了基于随机微分方程的开源软件可靠性模型，并根据总逾期软件维护工作量找到开源软件的最佳升级时间；Singh 等[5]为了适应故障跟踪系统故障报告现象的不规则状态和噪声的不规则波动，应用伊藤型随机微分方程建立了故障跟踪系统的可靠性模型；Zhang 等[6]针对故障检测率可能会受到某些因素的影响，并且可能会随着时间的推移在某个时间点发生变化这一问题，提出了具有变化点的随机微分方程软件可靠性模型来精确反映故障分布的变化；Tamura 等[7]考虑云计算中网络流量和大数据，提出了一个具有二维 Wiener 过程的跳跃扩散模型来评估云软件的稳定性。Fang 等[8]采用随机微分方程对软件故障检测过程进行置信区间估计，为软件开发人员和测试人员进行软件开发和软件质量控制提供了有益的信息；Wang 等[9]考虑软件调试过程中主观和客观的影响、故障排除的难度和复杂性、故障之间的依赖关系、软件测试不同阶段的变化、测试进度等因素，考虑到软件调试过程中故障引入率的不规律变化，提出了一种不完全软件调试模型；Chatterjee 等[10]针对开源软件开发了基于随机微分方程的软件可靠性增长模型，并考虑到故障分类的影响，制定了多版本的软件可靠性增长模型，并通过平衡可靠性水平最大化和产品开发成本

最小化来获得产品的最佳发布日期；Chatterjee 等[11]考虑软件故障的严重程度以及各故障被修复的优先级不同等因素，建立了一种基于随机微分方程的软件可靠性模型，将故障检测与故障发生的严重程度联系在一起，并将纠正故障的过程与缺陷修复的优先级联系在一起；Singh 等[12]考虑软件故障修复、功能增强和添加新功能等情况，给故障排除率带来了不确定性，使用三维维纳过程定义了三种波动类型，提出了一个统一的模型框架。综上，目前的随机微分方程可靠性模型都是结合经典模型和随机微分方程对总故障数据建模，但对复杂的系统软件，未考虑多组件叠加模型的情况。

本文针对提出的想法，基于各组件的故障数据建立叠加随机微分方程可靠性模型来估计总故障数据。其余部分主要内容如下：第二节主要介绍了经典 NHPP 类软件可靠性模型，以及叠加的随机微分方程可靠性模型的框架；第三节针对真实数据进行数值案例分析；最后一节给出结论。

2. 随机微分方程可靠性模型

2.1. 软件可靠性模型

软件可靠性模型是一种随机过程模型，将软件可靠性与失效时间、故障数等相关的量联系在一起。目前软件可靠性模型有多种，其中非齐次泊松过程(NHPP)类模型应用广泛。

NHPP 类软件可靠性增长模型的假设如下[13]：

1) 软件失效过程满足非齐次泊松过程；

2) 系统剩余的缺陷引起软件的失效；

3) 排除过程中不会引入新的缺陷。

记 $N(t)$ 为 $(0,t]$ 时间段内检测出的累计故障数，$\{N(t),t\geq 0\}$ 是一个独立增量过程；$m(t)$ 表示到 t 时刻累计故障数的期望值，$E[N(t)]=m(t)$；$\lambda(t)$ 为失效强度函数，表示单位时间内发现的故障数。有上述假设可知：

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

其中 a 表示软件预期的故障总数，$b(t)$ 为故障检测率函数，表示 t 时刻软件中潜伏的每个故障被检测到的概率。求解微分方程后可得：

$$m(t)=\mathrm{e}^{-\int_0^t b(u)\mathrm{d}u}\left[m(0)+a\int_0^t b(u)\mathrm{e}^{\int_0^t b(u)\mathrm{d}u}\mathrm{d}u\right] \tag{2}$$

令 $m(0)=0$，可简化均值函数为：

$$m(t)=a\mathrm{e}^{-\int_0^t b(u)\mathrm{d}u}\left[m(0)+a\int_0^t b(u)\mathrm{e}^{\int_0^t b(u)\mathrm{d}u}\mathrm{d}u\right] \tag{3}$$

对于多组件构成的软件系统，可利用各组件的故障数据来建立叠加模型。考虑软件包含 k 个组件的情况，需要建立可叠加的可靠性模型，其基本假设为：

1) 软件包含 k 个组件(子系统、对象或其他)；

2) 对象 i 的失效过程服从 NHPP，$\{N_i(t),t\geq 0\}$，其均值函数为 $m_i(t)$；

3) $\{N_i(t),t\geq 0\}(i=1,2,\cdots,k)$ 相互独立；

4) 软件的累积故障数为 $N(t)$，可知：

$$N(t)=\sum_{i=1}^{k}N_i(t) \tag{4}$$

其均值函数为：$m(t)=\sum_{i=1}^{k}m_i(t)$。

对公式(4)中的故障检测率$b(t)$取不同的形式，可以得到不同的 NHPP 模型，经典的模型如下：

1) 当$b(t)=t$时，得到经典的 GO 模型，其均值函数为：

$$m(t)=\sum_{i=1}^{k}a_i\left(1-\mathrm{e}^{-b_i t}\right) \tag{5}$$

2) 当$b(t)=\dfrac{b^2 t}{1+bt}$时，得到经典的 DSS 模型，其均值函数为：

$$m(t)=\sum_{i=1}^{k}a_i\left(1-(1+b_i)\mathrm{e}^{-b_i t}\right) \tag{6}$$

2.2. 可叠加的随机微分方程软件可靠性模型

考虑故障追踪系统中故障报告的随机性、不规律性以及各种不确定因素，故障检测过程可以看作是一个随机过程，用随机微分方程来建立相应的软件可靠性模型[5]。由前文的假设可以得到：

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t}=r(t)\left[a-N(t)\right] \tag{7}$$

其中$r(t)$可能含有不规则波动，因此可以得到：$r(t)=b(t)+\delta\gamma(t)$，$\delta$是一个表示不规则波动的正常量，$\gamma(t)$为标准的高斯白噪声。

将$r(t)$带入公式(7)可以得到：

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t}=\left[b(t)+\delta\gamma(t)\right]\left[a-N(t)\right] \tag{8}$$

公式(7)可以被扩展成下面的伊藤类随机微分方程：

$$\mathrm{d}N(t)=\left[b(t)-\frac{1}{2}\delta^2\right]\left[a-N(t)\right]\mathrm{d}t+\delta\left[a-N(t)\right]\mathrm{d}W(t) \tag{9}$$

其中$W(t)$表示与$\gamma(t)$有关的一维维纳过程，因此故障总数函数可以表示为：

$$N(t)=\sum_{i=1}^{k}N_i(t)=\sum_{i=1}^{k}a_i\left[1-\mathrm{e}^{-\int_0^t b_i(u)\mathrm{d}u-\delta_i W(t)}\right] \tag{10}$$

当$b(t)$分别取不同的取值时，可以得到基于 GO 模型的可叠加随机微分方程可靠性模型(GO_ASDE)以及基于 DSS 模型的可叠加随机微分方程可靠性模型(DSS_ASDE)：

$$N(t)=\sum_{i=1}^{k}a_i\left[1-\mathrm{e}^{-b_i t-\delta_i W(t)}\right] \tag{11}$$

$$N(t)=\sum_{i=1}^{k}a_i\left[1-(1+b_i t)\mathrm{e}^{-b_i t-\delta_i W(t)}\right] \tag{12}$$

因此，模型的均值函数分别为：

$$E\left[N(t)\right]=\sum_{i=1}^{k}a_i\left[1-\mathrm{e}^{-b_i t+\frac{1}{2}\delta_i^2 t}\right] \tag{13}$$

$$E\left[N(t)\right]=\sum_{i=1}^{k}a_i\left[1-(1+b_i t)\mathrm{e}^{-b_i t+\frac{1}{2}\delta_i^2 t}\right] \tag{14}$$

3. 数值案例

为了充分、有效地评估提出的改进模型的拟合和预测性能，本节使用了经典模型 GO 模型、DSS 模型以及本文改进模型 GO_ASDE、DSS_ASDE 的均值函数，使用真实数据集进行计算，对计算结果进行

对比分析。模型汇总见表 1：

Table 1. Model summary table
表 1. 模型汇总表

	模型名称	模型均值函数
Model 1	GO	$m(t)=a\left(1-\mathrm{e}^{-bt}\right)$
Model 2	DSS	$m(t)=a\left[1-(1+bt)\mathrm{e}^{-bt}\right]$
Model 3	GO_ASDE	$E\left[N(t)\right]=\sum_{i=1}^{k}a_i\left[1-\mathrm{e}^{-b_i t+\frac{1}{2}\delta_i^2 t}\right]$
Model 4	DSS_ASDE	$E\left[N(t)\right]=\sum_{i=1}^{k}a_i\left[1-(1+b_i t)\mathrm{e}^{-b_i t+\frac{1}{2}\delta_i^2 t}\right]$

3.1. 数据介绍

3.1.1. 数据集 1

数据集 1 的故障数据来自 Eclipse 故障跟踪系统(https://bugs.eclipse.org/bugs/)中的 AspectJ 项目，AspectJ 是 Eclipse 基金组织的开源项目，它是 Java 语言的一个面向切面编程(AOP)的实现，是最早、功能比较强大的 AOP 实现之一，可以在 Eclipse Foundation 开源中使用，既可以单独使用，也可以集成到 Eclipse 中。

AspectJ 的主要数据字段有：Compiler、IDE、LTWeaving、Docs、AJBrowser、AJDoc、Ant、Build、Library、Rutime 和 Testing 等，以月为单位进行数据整合。表 2 是各组件在 2005 年 1 月到 2020 年 11 月期间检测到的故障数(共 191 组数据，只展示部分数据)，其中 F1 表示组件 Compiler 的累计故障数，F2 表示组件 IDE 的累计故障数，F3 表示组件 LTWeaving 的累计故障数，剩余 8 个组件的故障数相对较少，F4 表示其他 8 个组件的累计故障数之和，F 为总累计故障数。

Table 2. The faults data of AspectJ (Dataset1)
表 2. AspectJ 故障数据(数据集 1)

t	F1	F2	F3	F4	F	t	F1	F2	F3	F4	F
1	13	2	0	3	18	21	556	33	32	47	668
2	16	2	0	3	21	22	576	33	38	51	698
3	36	5	0	5	46	23	584	33	42	52	711
4	48	9	0	5	62	24	585	33	42	53	713
5	64	9	0	6	79	25	588	33	43	54	718
6	88	9	0	7	104	26	600	34	43	54	731
7	95	9	0	9	113	27	603	34	43	55	735
8	163	15	0	13	191	28	604	34	43	55	736
9	218	16	0	18	252	29	604	34	43	55	736
10	265	18	0	22	305	30	606	34	43	55	738
11	339	19	0	24	382	31	624	34	43	55	756

Continued

12	366	20	0	27	413	32	624	34	43	55	756
13	382	23	1	27	433	33	625	34	43	55	757
14	401	23	2	29	455	34	687	42	49	64	842
15	420	23	4	29	476	35	702	42	49	67	860
16	431	26	5	30	492	36	705	42	49	67	863
17	491	31	9	37	568	37	715	42	49	68	874
18	511	31	15	38	595	38	730	43	51	71	895
19	527	31	19	38	615	39	756	44	53	74	927
20	544	31	26	46	647	40	765	44	53	75	937

3.1.2. 数据集 2

数据集 2 的数据来自 Eclipse 故障跟踪系统(https://bugs.eclipse.org/bugs/)中的 Babel 项目，Babel 是一个 JavaScript 编译器，使软件开发者能够以偏好的编程语言或风格来写作源代码，并将其利用 Babel 翻译成 JavaScript。

Babel 的主要数据字段有：Plugins、Website、EnglishStrings、Translation 和 Server，以月为单位进行数据整合，表 3 是各组件在 2008 年 1 月到 2016 年 12 月的期间检测到的故障数(共 108 组数据，只展示部分数据)。其中 F1 表示组件 Plugins 的累计故障数，F2 表示组件 Server 的累计故障数，F3 表示组件 Website 的累计故障数，F4 表示组件 EnglishStrings 和 Translation 的累计故障数之和，F 为总累计故障数。

Table 3. The faults data of AspectJ (Dataset2)
表 3. AspectJ 故障数据(数据集 2)

t	F1	F2	F3	F4	F	t	F1	F2	F3	F4	F
1	1	9	2	0	12	21	43	218	29	22	312
2	8	17	4	0	29	22	45	219	31	24	319
3	13	34	5	0	52	23	45	220	32	24	321
4	19	50	5	0	74	24	45	221	32	24	322
5	19	63	8	0	90	25	46	223	32	27	328
6	21	75	9	0	105	26	47	228	32	28	335
7	25	93	10	0	128	27	48	229	33	28	338
8	26	99	11	0	136	28	48	232	33	28	341
9	26	101	11	0	138	29	49	235	33	28	345
10	29	115	11	0	155	30	49	235	33	29	346
11	30	137	13	0	180	31	49	237	34	34	354
12	33	153	13	0	199	32	49	239	34	35	357
13	36	170	15	1	222	33	49	239	34	36	358
14	37	177	15	2	231	34	49	239	34	36	358
15	39	192	20	6	257	35	49	240	34	36	359

Continued

16	39	201	22	9	271	36	49	243	34	37	363
17	39	203	25	9	276	37	49	243	34	39	365
18	40	215	28	17	300	38	50	243	35	42	370
19	40	216	28	19	303	39	51	244	35	44	374
20	43	217	29	20	309	40	51	244	35	44	374

3.2. 模型评价指标

为了比较各模型的性能，选择以下评价指标进行衡量[14]：AIC，MSE。

1) AIC 信息准则：AIC 是衡量模型优良性的一种标准，其定义为 $\mathrm{AIC}=2K-2\log(L)$，其中 K 为参数数量，L 为极大似然值。当误差服从正态分布时，可得：

$$\mathrm{AIC}=2K+m\log\left(\frac{\mathrm{RSS}}{m}\right) \tag{15}$$

其中 m 为样本容量，RSS 为残差平方和。

2) MSE：均方误差，更小的 MSE 有更好的拟合效果。其定义为：

$$\mathrm{MSE}=\sum_{j=1}^{m}\frac{\left(m(t_j)-m_j\right)^2}{k} \tag{16}$$

其中 $m(t_j)$ 表示到时刻 t_j 系统累计故障数的估计值，m_j 表示到时刻 t_j 系统累计观测到的故障数。

3.3. 参数估计结果

3.3.1. 数据集 1 结果

使用表 2 中 AspectJ 真实数据集对经典模型和本文模型进行参数估计，参数估计结果及评价指标见表 4。

Table 4. Parameter estimation results for Dataset 1
表 4. 数据集 1 的参数估计结果

模型名称		参数估计值			MSE	AIC
Model 1	GO	$a=3115$	$b=0.0103$		5033	1632
Model 2	DSS	$a=2599$	$b=0.0310$		11043	1782
Model 3	GO_ASDE	$a_1=2091$	$b_1=0.1868$	$\delta_1=0.5899$	650	996
		$a_2=144.2$	$b_2=0.0484$	$\delta_2=0.2761$		
		$a_3=781.5$	$b_3=0.1668$	$\delta_3=0.5730$		
		$a_4=590.6$	$b_4=0.0541$	$\delta_4=0.3451$		
Model 4	DSS_ASDE	$a_1=1859$	$b_1=0.0341$	$\delta_1=0.0023$	1922	1068
		$a_2=122.2$	$b_2=0.0298$	$\delta_2=1e{-}8$		
		$a_3=289.2$	$b_3=0.0315$	$\delta_3=0.1066$		
		$a_4=331.3$	$b_4=0.0248$	$\delta_4=0.0287$		

从表 4 中可以看出：对于经典模型，GO 模型的 MSE 和 AIC 要小于 DSS 模型，更适合 AspectJ 的故障数据；针对 GO 模型来说，本文改进后的叠加随机微分方程模型的效果要优于经典 GO 模型。从结果可以看出利用随机微分方程建立的叠加模型的效果要比经典模型要好，这也说明了把故障检测过程可以看作是一个随机过程，用随机微分方程来建立相应的软件可靠性模型是合理且有效的。四个模型的累计故障数的拟合结果如图 1 所示，各模型的 RE 图如图 2。

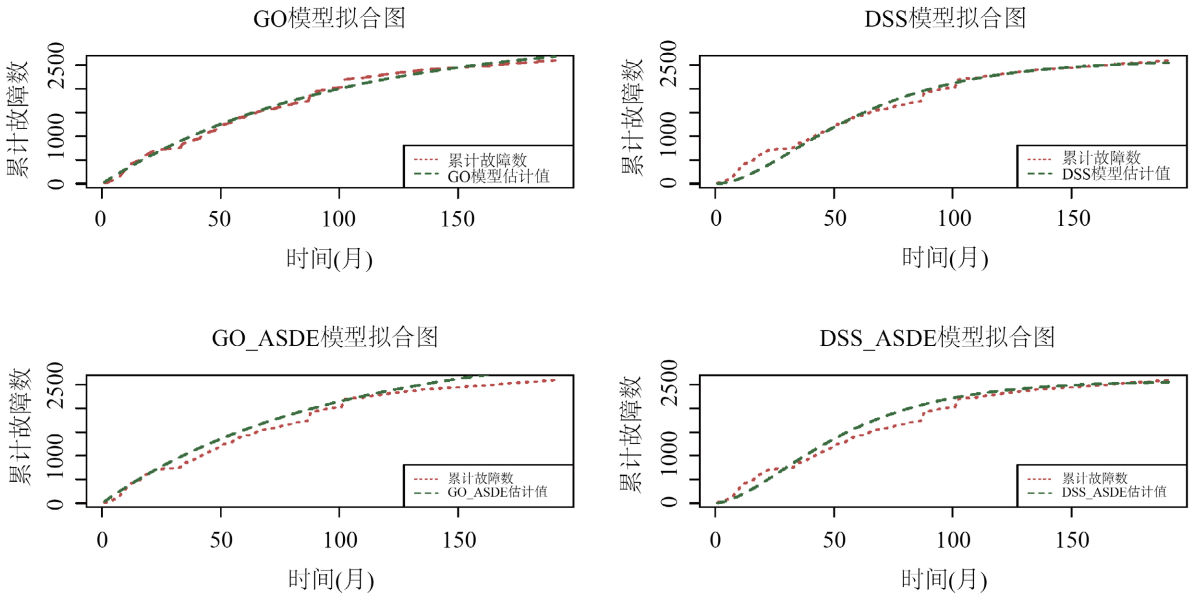

Figure 1. Fitted plots of the cumulative number of failures for each model
图 1. 各模型的累计故障数拟合图

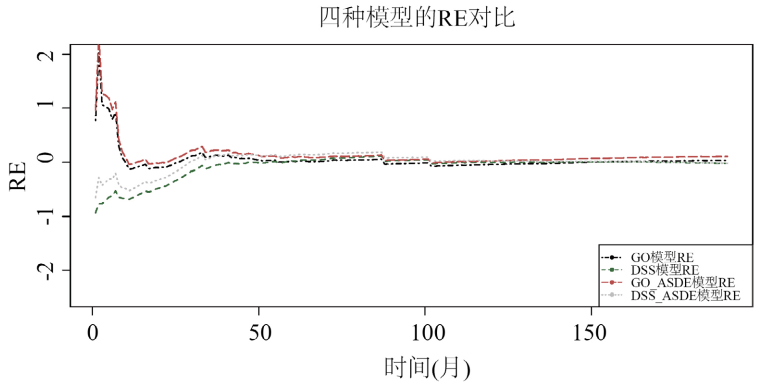

Figure 2. RE comparison chart for each model
图 2. 各模型的 RE 对比图

3.3.2. 数据集 2 结果

使用数据集 2 的 Babel 真实数据进行估计，参数估计结果及模型的评价指标结果如表 5 所示。

对于经典模型，GO 模型更适合该数据，MSE 和 AIC 均小于 DSS 模型。同时改进后的叠加随机微分方程 GO 模型、DSS 模型的效果均优于经典的 GO 模型、DSS 模型。从结果可以看出把故障检测过程看作随机过程来建立软件可靠性模型是合理且有效的。四个模型的累计故障数的拟合结果如图 3 所示，各模型的 RE 图如图 4。

Table 5. Parameter estimation results for Dataset 2
表 5. 数据集 2 的参数估计结果

模型名称		参数估计值			MSE	AIC
Model1	GO	$a=632.6$	$b=0.0231$		961	746
Model2	DSS	$a=604.0$	$b=0.0514$		4029	901
Model3	GO_ASDE	$a_1=94.5$	$b_1=0.0231$	$\delta_1=0.0318$	129	455
		$a_2=329$	$b_2=0.1133$	$\delta_2=0.3859$		
		$a_3=89.4$	$b_3=0.0127$	$\delta_3=0.0016$		
		$a_4=190$	$b_4=0.0081$	$\delta_4=0.0299$		
Model4	DSS_ASDE	$a_1=88.00$	$b_1=0.0545$	$\delta_1=0.00001$	190	486
		$a_2=290.5$	$b_2=0.1250$	$\delta_2=0.0029$		
		$a_3=71.36$	$b_3=0.0427$	$\delta_3=0.0457$		
		$a_4=119.5$	$b_4=0.0395$	$\delta_4=0.0888$		

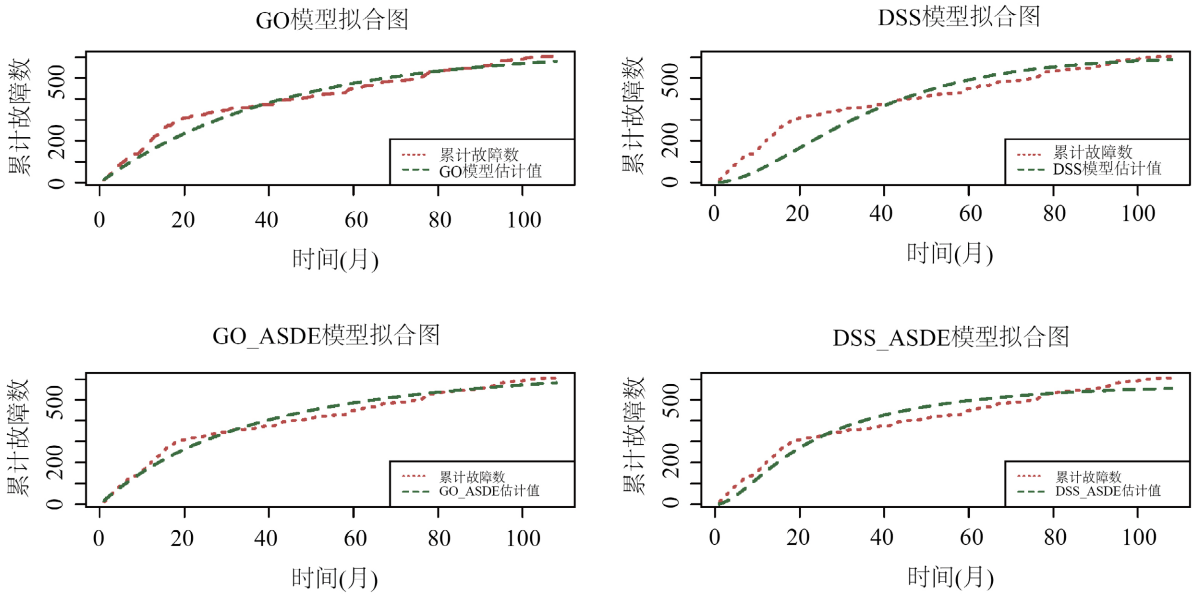

Figure 3. Fitted plots of the cumulative number of failures for each model
图 3. 各模型的累计故障数拟合图

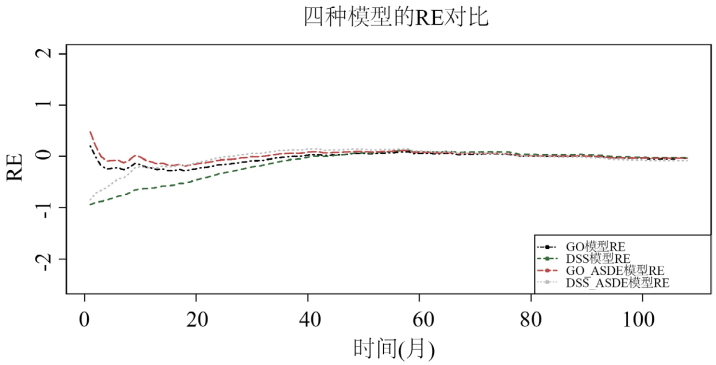

Figure 4. RE comparison chart for each model
图 4. 各模型的 RE 对比图

4. 结论

在多组件的软件系统中，软件系统的可靠性估计十分重要，估计各组件的可靠性也是不可或缺的。实际中，软件可靠性分析所需的故障数据大多来自软件的故障跟踪系统，该系统上报告的软件故障存在随机性和不确定性，会导致故障检测率会受一些随机影响。因此本文针对多组件软件系统，考虑软件故障报告的随机性情况，利用组件的故障数据来建立软件可靠性叠加模型，将经典的 NHPP 类软件可靠性模型改进为叠加的随机微分方程软件可靠性模型，度量软件系统可靠性的同时也可以对各个组件的可靠性做一个度量。从 Eclipse 故障跟踪系统上获得了 AspectJ 项目和 Bebel 项目的故障数据，通过分析对比发现叠加的随机微分方程可靠性模型的效果优于经典的 NHPP 类软件可靠性模型，该模型度量了每个组件的可靠性情况以及整体的可靠性，且拟合效果更好。

基金项目

国家自然科学基金资助项目(71901078)；贵州省电力大数据重点实验室(黔科合计 Z 字[2015] 4001)。

参考文献

[1] Wang, L., Hu, Q. and Liu, J. (2016) Software Reliability Growth Modeling and Analysis with Dual Fault Detection and Correction Processes. *IIE Transactions*, **48**, 359-370. https://doi.org/10.1080/0740817X.2015.1096432

[2] Kapur, P., Goswami, D. and Bardhan, A. (2007) A General Software Reliability Growth Model with Testing Effort Dependent Learning Process. *International Journal of Modelling and Simulation*, **27**, 340-346. https://doi.org/10.1080/02286203.2007.11442435

[3] Jain, M., Manjula, T. and Gulati, T. (2014) Imperfect Debugging Study of SRGM with Fault Reduction Factor and Multiple Change Point. *International Journal of Mathematics in Operational Research*, **6**, 155-175. https://doi.org/10.1504/IJMOR.2014.059526

[4] Tamura, Y. and Yamada, S. (2009) Optimisation Analysis for Reliability Assessment Based on Stochastic Differential Equation Modelling for Open Source Software. *International Journal of Systems Science*, **40**, 429-438. https://doi.org/10.1080/00207720802556245

[5] Singh, V., Kapur, P.K. and Tandon, A. (2010) Measuring Reliability Growth of Open Source Software by Applying Stochastic Differential Equations. *Proceedings of the* 2010 *Second World Congress on Software Engineering*, 19-20 December 2010, Hubei, 115-118. https://doi.org/10.1109/WCSE.2010.149

[6] Zhang, N., Cui, G., Shu, Y. and Liu, H. (2014) Stochastic Differential Equation Software Reliability Growth Model with Change-Point. *High Technology Letters*, **20**, 383-389.

[7] Tamura, Y. and Yamada, S. (2015) Reliability Analysis Based on a Jump Diffusion Model with Two Wiener Processes for Cloud Computing with Big Data. *Entropy*, **17**, 4533-4546. https://doi.org/10.3390/e17074533

[8] Fang, C.-C. and Yeh, C.-W. (2016) Effective Confidence Interval Estimation of Fault-Detection Process of Software Reliability Growth Models. *International Journal of Systems Science*, **47**, 2878-2892. https://doi.org/10.1080/00207721.2015.1036474

[9] Wang, J. (2017) An Imperfect Software Debugging Model Considering Irregular Fluctuation of Fault Introduction Rate. *Quality Engineering*, **29**, 377-394. https://doi.org/10.1080/08982112.2017.1310229

[10] Chatterjee, S., Chaudhuri, B. and Bhra, C. (2020) Optimal Release Time Determination in Intuitionistic Fuzzy Environment Involving Randomized Cost Budget for SDE-Based Software Reliability Growth Model. *Arabian Journal for Science and Engineering*, **45**, 2721-2741. https://doi.org/10.1007/s13369-019-04128-7

[11] Chatterjee, S., Chaudhuri, B. and Bhra, C. (2021) Optimal Release Time Determination via Fuzzy Goal Programming Approach for SDE-Based Software Reliability Growth Model. *Soft Computing*, **25**, 3545-3564. https://doi.org/10.1007/s00500-020-05385-7

[12] Singh, O., Anand, A. and Singh, J. (2021) SDE Based Unified Scheme for Developing Entropy Prediction Models for OSS. *International Journal of Mathematical, Engineering and Management Sciences*, **6**, 207. https://doi.org/10.33889/IJMEMS.2021.6.1.013

[13] 张策, 孟凡超, 考永贵, 等. 软件可靠性增长模型研究综述[J]. 软件学报, 2017, 28(9): 2402-2430.

[14] 陈静, 杨剑锋, 王喜宾, 等. NHPP 类开源软件可靠性增长模型的极大似然估计[J]. 广西大学学报(自然科学版), 2022, 47(1): 174-184.