

基于未确知理论的软件可靠性建模^{*}

张永强⁺, 孙胜娟

(河北工程大学 信息与电气工程学院, 河北 邯郸 056038)

Software Reliability Modeling Based on Unascertained Theory

ZHANG Yong-Qiang⁺, SUN Sheng-Juan

(School of Information and Electrical Engineering, Hebei University of Engineering, Handan 056038, China)

+ Corresponding author: Phn: +86-310-7428733, Fax: +86-310-7428746, E-mail: yqzhang@hebeu.edu.cn, <http://www.hebeu.edu.cn/>

Zhang YQ, Sun SJ. Software reliability modeling based on unascertained theory. *Journal of Software*, 2006,17(8):1681-1687. <http://www.jos.org.cn/1000-9825/17/1681.htm>

Abstract: In this paper, unascertained mathematical theory is applied to the study of software reliability modeling. It is used to study software fault process, describe software failure character and get software reliability parameters. Finally, a software reliability model based on unascertained theory is proposed in this paper. The new model changes the traditional modeling thought, and brakes away from the statistical distribution assumption about the variety of failure rate in the traditional software reliability modeling process. It has better applicability, which improves the incongruence of model application to some extent.

Key words: software reliability; reliability model; modeling; applicability; unascertained mathematical

摘要: 将未确知理论应用于软件可靠性建模研究,采用其分析软件故障过程,用未确知数学描述软件失效特征计算软件可靠性参数,并在此基础上构建了一个基于未确知数学理论的软件可靠性模型.新模型改变了传统的建模思路,跳出了传统软件可靠性建模过程中关于失效强度变化的各种统计分布假设的束缚,具有较好的适用性,改善了模型应用中的不一致性问题.

关键词: 软件可靠性;可靠性模型;建模;适用性;未确知数学

中图法分类号: TP311 文献标识码: A

随着软件应用的日益广泛及重要性的不断增强,人们对软件质量的要求也越来越高.可靠性作为衡量软件质量的重要特性,其定量评估和预测已成为人们关注和研究的焦点.软件可靠性模型作为可靠性评测的核心和关键,可用于软件生命周期的不同阶段,定量地估计和预测软件可靠性行为.一个好的可靠性模型可以准确评估和预测软件可靠性行为,这对于软件资源分配、软件市场决策有着重要的意义^[1].我们可以看到,软件可靠性模型这一领域的研究在 20 世纪 70 年代获得较大发展后,很多可靠性模型已经投入实用.可以说,软件可靠性模型已经从研究阶段发展到了工程阶段.但是,面对软件自身及其开发过程日益复杂的情形,它仍然呈现出其自身的不足.

^{*} Supported by the National Natural Science Foundation of China under Grant No.60573088 (国家自然科学基金); the Natural Science Foundation of Hebei Province of China under Grant No.603407 (河北省自然科学基金)

Received 2005-07-20; Accepted 2005-12-13

首先,在软件可靠性建模方面.传统的软件可靠性模型一般都是基于演绎的建立在经典的概率统计基础之上的.人们为建立演绎模型不得不作出一些精确、无歧义的假设,以便将软件可靠性增长行为这个复杂的物理过程进行抽象和简化,从而最终建立预测模型^[2].据此也不难想象,为何传统软件可靠性增长模型会存在所谓模型应用不一致性问题.模糊数学的发展,深化了软件可靠性工程中的模糊现象的认识^[3],但它只是概率方法的合理补充,并未从根本上解决问题.人工神经网络在软件可靠性估测中显示出良好的一致性,准确度高,但它对建模数据要求也高,且存在学习不足和过拟合的问题.另外,它的模型过于复杂,而且要花大量时间培训,这就降低了它的实用性.

其次,在软件可靠性模型应用方面,存在严重的应用不一致性.所谓不一致性就是这些模型只是有时非常精确,换一个环境,应用到其他工程项目中,其精度就不能满足要求了,亦即软件可靠性模型的精度不高、适用性不好.产生软件可靠性模型应用不一致性的根源在于模型建立的前提假设各不相同,每种可靠性模型都有关于故障过程中失效强度变化规律的假设,这样,只有在某一工程项目的失效强度是按这些假定规律变化的时候,模型的精度才会很高;一旦工程项目的失效强度不按这些规律变化时,模型的精度就会很糟.目前看来,如果又是根据一定的假设提出新模型,只会给软件开发人员对模型应用不一致性带来更大的困惑.因此,改变传统可靠性建模思路,采用新的观点、方法和新的数学工具来研究软件故障过程,摆脱建模理论假设的束缚,不受其制约,才有可能为建立新的、较普适的软件可靠性模型提供可能.

一个软件在完成未使用前,其程序中的隐患(bug),即潜在的程序错误数为一个确定值,但由于测试者因条件限制认识不清,无法作出正确回答,因此,它对测试者来说是不确定的.此时,程序隐患数是一个未确知有理数;一个软件的所有输入情况是有限的、确定的,但测试者无法穷举其所有情况组合,它对测试者来说也是不确定的.此时,所有输入情况是一组未确知信息;由于条件限制,测试者无法确切列举一个软件各种操作组合,因此,用户的操作集合也是一组未确知信息.因此我们可以认为,软件故障过程是一种未确知系统,正是未确知数学理论所研究的对象^[4].

鉴于此,我们进行了相关探索,将未确知理论应用到软件可靠性建模的研究中去,采用其研究软件故障过程,用未确知数描述软件失效特征,并依据未确知数学方法求得软件可靠性参数,在此基础上构建了一个基于未确知数学理论的软件可靠性模型.该模型跳出了传统软件可靠性建模过程中关于失效强度变化的各种统计分布假设的束缚,具有较高的预计精度和较好的适用性,改善了模型应用中的不一致性问题.

1 未确知理论相关知识

未确知数学理论主要研究表达和处理未确知信息的方法.未确知信息是不同于模糊信息、随机信息和灰信息的一种新的不确定性信息,这种信息的特点在于:它的不确定性主要不是客观的,而是决策者主观的、认识上的不确定性.这种由于主观或客观原因,决策者不能完全认识事物的真实状态或确定的数量关系,在心目中产生的主观的、认识上的不确定性,称为未确知性.具有这种未确知性的信息成为未确知信息.

未确知数学信息处理方法最大的特点是:保留所有已知信息,直接参与定量运算,可使积累误差减到最小.而且,除了原始数据以外,没有任何人为假定,可最大程度地忠实于所给出的信息.未确知有理数是最简单、应用最广泛的表达未确知信息的方法,下面我们给出未确知有理数的数学定义^[5]:

对于任意闭区间 $[a, b], a = x_1 < x_2 < \dots < x_n = b$, 若函数 $\varphi(x)$ 满足

$$\varphi(x) = \begin{cases} \alpha_i, & x = x_i \quad (i = 1, 2, \dots, n) \\ 0, & \text{others} \end{cases},$$

且 $\sum_{i=1}^n \alpha_i = \alpha, 0 < \alpha \leq 1$, 则称 $[a, b]$ 和 $\varphi(x)$ 构成一个 n 阶未确知有理数,记作 $[[a, b], \varphi(x)]$,称 $\alpha, [a, b]$ 和 $\varphi(x)$ 分别为该未确知有理数的总可信度、取值区间和可信度分布函数.

设 A 为上式表达的未确知有理数,称一阶未确知有理数

$$E(A) = \left[\left[\frac{1}{\alpha} \sum_{i=1}^k x_i \alpha_i, \frac{1}{\alpha} \sum_{i=1}^k x_i \alpha_i \right], \varphi(x) \right]$$

为未确知有理数 A 的数学期望,其中

$$\varphi(x) = \begin{cases} \alpha, & x = \frac{1}{\alpha} \sum_{i=1}^k x_i \alpha_i, \\ 0, & \text{others} \end{cases}$$

也称 $E(A)$ 为未确知期望,简称期望或均值^[5].

2 软件可靠性未确知模型

2.1 建模思想

对可以获得的部分系统输出数据即失效数据进行分析,将其蕴涵的系统失效特征用未确知有理数表达出来,构造软件故障过程的等价系统,从而完成对软件系统失效行为的刻画,并依据所建模型完成对系统未来失效行为的预测.

2.2 基本假设与数据要求

基本假设:程序测试环境与预期使用环境相同.

这是软件可靠性建模中的标准假设,确保在某一特定环境下使用数据采集进行模型评价的正确性^[2].也就是说,该假设应能保证数据采集得到的失效数据适用于软件可靠性的度量.

实现此模型的数据需求:软件失效间隔时间分别为 x_1, x_2, \dots, x_n , 失效时间分别为 t_1, t_2, \dots, t_n , 其中: $x_i = t_i - t_{i-1}$, $i = 1, 2, \dots, n, t_0 = 0$.

2.3 基本公式

软件可靠性未确知模型(software reliability unascertained model)将第 i 次失效间隔时间看作是一个未确知有理数 x_i , 即时刻 t 的平均失效间隔时间为 $E(x_i)$, 其中 t 是第 $i-1$ 次失效到第 i 次失效之间的任意时刻.

以第 $i-1$ 次失效为起点到第 i 次失效发生的时间是一个未确知有理数,记为

$$x_i = [[x_{\min}, x_{\max}], \varphi(x_i)],$$

其中

$$x_{\min} = \min(x_{i-1}, x_{i-2}, \dots, x_{i-m}), \quad x_{\max} = \max(x_{i-1}, x_{i-2}, \dots, x_{i-m}).$$

m 为参与运算的失效数据个数、一个待定常数.对确认测试,可取 $i-1$;对可靠性测试,我们认为早期数据对预测未来行为作用很小,现时失效间隔数据比许久之前观测的失效间隔数值更好地预测未来, m 一般取 8~10. $\varphi(x_i)$ 是 x_i 的可信度密度分布函数.我们定义 $\varphi(x_i)$, 若 x_j 邻域内 x_k 多,则认为 x_i 取值 x_j 的可信度大;反之,则认为 x_i 取值 x_j 的可信度很小,其中 $j, k = i-m, \dots, i-1$, 且 $j \neq k$. 具体定义为

$$\varphi(x_i) = \begin{cases} \frac{\xi_j}{\xi}, & x_i = x_j, \\ 0, & \text{others} \end{cases}$$

其中: ξ_j 表示 x_j 邻域 $|x_j - x_k| \leq r$ 中包含 $x_k (j \neq k)$ 的个数, $\xi = \sum_{j=i-m}^{i-1} \xi_j$; r 为 x_j 的邻域半径,可凭经验由失效数据估计出来.应注意,在此过程中, r 是不断调整的.

x_i 的数学期望 $E(x_i)$ 为时刻 t 的平均失效间隔时间 MTBF, 其中 t 是第 $i-1$ 次失效到第 i 次失效之间的任一时刻.在稳定使用软件且不对软件作任何修改的情况下,软件的失效强度为一个常数,其值为 $\frac{1}{MTBF}$ ^[6].因此,在

两次失效间隔时间内,失效强度是常数 λ , 其值为 $\frac{1}{E(x_i)}$, 即

$$\lambda = \frac{1}{E(x_i)},$$

其中

$$E(x_i) = \sum_{j=i-m}^{i-1} x_j \frac{\xi_j}{\xi}, \quad \xi = \sum_{j=i-m}^{i-1} \xi_j.$$

以第 $i-1$ 次失效为起点到第 i 次失效发生的时间 x_i 的密度函数为

$$f(x_i) = \frac{1}{E(x_i)} \exp\left\{-\frac{1}{E(x_i)} x_i\right\},$$

其分布函数为

$$F(x_i) = 1 - \exp\left\{-\frac{1}{E(x_i)} x_i\right\},$$

其可靠性函数为

$$R(x_i) = \exp\left\{-\frac{1}{E(x_i)} x_i\right\},$$

其中

$$E(x_i) = \sum_{j=i-m}^{i-1} x_j \frac{\xi_j}{\xi}, \quad \xi = \sum_{j=i-m}^{i-1} \xi_j.$$

3 模型评价

软件可靠性模型所要解决的问题有两个:改进软件开发过程和软件可靠性的度量.相应地,针对可靠性模型及其应用展开的研究主要集中在两个方面:关于软件可靠性早期预测模型的研究和关于软件可靠性预测模型的研究^[7].其中,早期预测模型是指在不知失效数据的情况下,根据软件产品及其开发过程度量预测软件可靠性.这种模型对于改进软件开发过程、指导软件测试、提高软件可靠性具有重要意义;可靠性预测模型着眼于未来,对软件的可靠性进行预计,预计软件当前失效强度、下次失效时间等^[8].本文所建立的软件可靠性未确知模型为软件可靠性预测模型,因此,对模型的评价从预测能力和适用性两个方面来进行.

基于未确知理论的软件可靠性模型根据失效时序数据进行建模,不对软件故障过程作任何统计规律的分布假设,即不假定失效强度变化服从特定的统计规律,更真实地描述了软件失效特征,理论上具有较高的预计精度和较好的适用性.下面,我们用常用的几种模型评价准则对它的预测能力和适用性进行评价.

3.1 短期预测能力比较

模型短期预测能力评价的依据是美国海军舰队计算机程序设计中心(U.S. Navy Fleet Computer Programming Center)的海军战术数据系统 NTDS(Naval Tactical Data System)开发过程(前 26 组数据)和测试过程(27~30 组数据)中的错误统计数据^[9].它是经过分析和整理的经典数据集,符合模型评价的标准假设,具有较高的精度.

针对软件失效间隔时间序列 NTDS,应用传统软件可靠性模型 G-O 模型、J-M 模型与新模型短期预测能力进行比较.目前已经推荐出几个有显著性检验的变量标准.在此,应用 Jason 在 1999 年提出的度量元(SRE)来度量模型的短期(即一步)预测能力,期望从模型预计值和实际观测值的 SRE 来判断新模型与 G-O 模型和 J-M 模型的一步预测能力.

计算度量元 SRE 的公式如下:

$$SRE = \frac{\sum_{i=1}^{n-1} \frac{|x_r(i+1) - x_p(i+1)|}{x_r(i+1)}}{n-1}.$$

使用到 $t_i(i>1)$ 时刻的失效数据建立模型,计算下一失效间隔时间 x_{i+1} 和下次出现失效的时间 $t_{i+1}(i>1)$.其中, $t_{i+1} = t_i + x_{i+1}(i>1)$.应注意到,在此过程中,用以计算拟合情况时,模型参数是不断进行调整的.其中, $x_r(i+1)$ 表示实际观察到的下一失效间隔时间, $x_p(i+1)$ 表示应用前 i 个故障数据建立的模型预测的下一失效间隔时间.SRE 值越小,说明模型短期预测能力越强,一步向前预测越准确.

对 G-O 模型、J-M 模型和新模型分别进行计算:以前 i 个失效间隔数据为建模数据,预测第 $i+1$ 个失效的间隔时间.对于 NTDS 中的数据,得到 27~30 个失效数据的一步结果及短期预测能力度量元 SRE 的值,见表 1.

Table 1 Prediction result

表 1 预测结果

Serial number	U-M	J-M	G-O
27	5.025	28.1	21.607 8
28	9.125	34.8	35.758 5
29	12.6	45.6	46.942 2
30	15.937 5	66.4	50.319 7
SRE	0.642 2	2.528 6	2.123 4

比较短期预测能力度量元, $SRE_{\text{新}} < SRE_{G-O} < SRE_{J-M}$, 说明基于未确知理论的新模型的短期预测能力要优于 G-O 模型和 J-M 模型.

3.2 适用性评价

将新模型与 J-M 模型和 G-O 模型应用于 4 组工程数据 NTDS, SYS1, SYS2, SYS3, 其中:NTDS 为上文使用的美国海军战术数据系统开发过程和测试过程中收集的数据, SYS1, SYS2, SYS3 来自 Musa 公开发表的标准数据集^[2], 是从实际工程项目中收集来的, 经过分析和整理, 具有很高的精度, 专门为比较、验证和评价软件可靠性模型而设计. 表 2~表 5 给出了 3 种模型对 4 组数据的适用性分析结果. 对所用模型的精确度(即预计精度用序列似然度来度量)、模型偏差、偏差趋势等作了评价, 各项数值在表中给出, 同时排出了优劣顺序, 在圆括号内指出对各项名次, 采用传统的模型选择方法, 即名次累加排出总体名次.

Table 2 Model applicability to data NTDS

表 2 模型对数据 NTDS 适用性

Criterion	U-M	J-M	G-O
$-\log(PL)$	52.759 (3)	51.757 (1)	52.636 (2)
Model bias	0.201 (1)	0.269 (3)	0.252 (2)
Bias tendency	0.323 (2)	0.310 (1)	0.342 (3)
Fit degree	0.155 (1)	0.264 (2)	0.266 (3)
Model noise	0.613 (1)	1.695 (3)	1.360 (2)
Overall place	(1)	(2)	(3)

Table 3 Model applicability to data SYS1

表 3 模型对数据 SYS1 适用性

Criterion	U-M	J-M	G-O
$-\log(PL)$	857.052 (1)	1 115.515 (3)	858.947 (2)
Model bias	0.346 (2)	0.479 (3)	0.168 (1)
Bias tendency	0.063 (1)	0.153 (3)	0.096 (2)
Overall place	(1)	(3)	(2)

Table 4 Model applicability to data SYS2

表 4 模型对数据 SYS2 适用性

Criterion	U-M	J-M	G-O
$-\log(PL)$	631.672 (2)	715.913 (3)	536.074 (1)
Model bias	0.263 (2)	0.309 (3)	0.143 (1)
Bias tendency	0.053 (1)	0.105 (3)	0.064 (2)
Overall place	(2)	(3)	(1)

Table 5 Model applicability to data SYS3

表 5 模型对数据 SYS3 适用性

Criterion	U-M	J-M	G-O
$-\log(PL)$	1 000.365 (1)	1 672.935 (3)	1 002.891 (2)
Model bias	0.188 (2)	0.253 (3)	0.084 (1)
Bias tendency	0.053 (1)	0.104 (3)	0.065 (2)
Overall place	(1)	(3)	(2)

对于数据 NTDS,新模型表现出了很好的性能,在模型偏差、拟合度、模型噪声 3 个指标上都优于 J-M 模型和 G-O 模型,J-M 模型也表现出较好的性能,在预计精度和偏差趋势两个指标上都是最优的,而 G-O 模型则表现较差;对于 SYS 系列数据,新模型也表现出很好的性能,尤其是对数据 SYS1 和 SYS3,新模型在预计精度和偏差趋势两个主要指标上都是最优的,综合排名第一位,G-O 模型也表现出较好的性能,对数据 SYS2 在预计精度和模型偏差两个指标上是最优的,综合排名也是第一位,而 J-M 模型则表现较差。

从以上分析可以看出:G-O 模型对 SYS 系列数据具有较好的适用性,但是对数据 NTDS 适用性却很差;J-M 模型对数据 NTDS 适用性较好,但对 SYS 系列数据的适用性却很差;而新模型对这 4 组数据都具有较好的适用性.这是由于 SYS 系列工程数据的失效特性符合 G-O 模型关于失效强度按指数衰减的理论假设,而工程数据 NTDS 的失效特性不符合 G-O 模型关于失效强度变化的理论假设;工程数据 NTDS 的失效特性符合 J-M 模型关于失效强度按几何衰减的理论假设,SYS 系列工程数据的失效特性不符合 G-O 模型关于失效强度变化的理论假设.由于不对失效强度变化作任何统计规律的假设,新模型对这两种具有不同特性的工程数据都有较好的适用性.

4 结束语

本文将未确知理论应用到软件可靠性模型研究中,从未确知理论的观点研究软件失效机理和软件故障过程,采用未确知数学表达和处理软件故障过程中的不确定性信息,为软件可靠性建模提供了新的思路.建立了基于未确知数学理论的软件可靠性预计模型.由于新模型不对软件失效强度变化作任何统计规律的假定,依据历史失效数据用未确知有理数来表示下次失效间隔时间,并依据未确知期望求得此时的软件平均故障间隔时间和失效强度,其基本假设更接近软件实际故障过程,跳出了传统软件可靠性建模过程中关于失效强度变化的各种统计分布假设的束缚,有较高的预计精度和较好的适用性,改善了模型应用中的不一致性问题.但是,其中仍有些问题需要进一步研究,如将新模型应用于更多的工程数据,对新模型的适用性作进一步分析;从理论上证明新模型的预测能力和适用性优于经典模型,进而说明新模型的优越性具有普遍性等.总之,未确知理论为软件可靠性评估和软件可靠性建模提供了更完备的信息和建模手段,很值得我们做进一步的研究.

致谢 在此,我们向对本文的工作给予支持和建议的同行表示感谢;对国家自然科学基金和河北省自然科学基金的资助表示感谢.

References:

- [1] Kapur PK, Singh O, Mittal R. Software reliability growth and innovation diffusion models: An interface. *Int'l Journal of Reliability, Quality and Safety Engineering*, 2004,11(4):339-364.
- [2] Lyu MR, *et al.*, ed.; Liu XC, Zhong WY, *et al.*, Trans. Handbook of Software Reliability Engineering. Beijing: Publishing House of Electronics Industry, 1997. 79-99 (in Chinese).
- [3] Utkin LV, Gurov SV. A fuzzy software reliability model with multiple-error introduction and removal. *Int'l Journal of Reliability, Quality and Safety Engineering*, 2002,9(3):215-227.
- [4] Zhang YQ, Sun SJ. Study of Uncertain factors in software reliability with mathematical method. In: Proc. of the 9th Joint Int'l Computer Conf. 2003. 138-141.
- [5] Liu KD, Wu HQ, Pang YJ, Liu KZ, Gao ZQ. Mathematical Treatment and Application of Uncertain Information. Beijing: Science Press, 1999. 43-86 (in Chinese).
- [6] Lu MY. Study on software reliability parameters, *Journal of Beijing University of Aeronautics and Astronautics*, 2001,27(2): 241-244 (in Chinese with English abstract).
- [7] Kan SH; Wang ZY, Chen L, Yu Y, *et al.*, Trans. Metrics and Models in Software Quality Engineering. Beijing: China Machine Press, 2003. 147-148 (in Chinese).
- [8] 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.

