第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •

文章编号：1000—3428(2012)03—0051—03 文献标识码：A 中图分类号：TP301

分布式系统可靠性模型研究

杨春晖，熊 婧，李 冬

(工业和信息化部电子第五研究所软件质量工程研究中心，广州 510610)

摘 要：为描述分布式开发环境中的软件可靠性增长情况，提出一种分布式开发环境下软件系统的可靠性模型。在实际的软件可靠性数据集上，通过使用适应性度量和 AIC 准则，对所提模型进行验证和评价。实验结果证明，该模型能捕获大量的可靠性成长曲线，具备较强的适用性和灵活性，且使用该模型所描述的分布式软件开发环境更符合实际情况。

关键词：分布式系统；软件测试；软件可靠性；可靠性模型

Research on Reliability Model for Distributed System

YANG Chun-hui, XIONG Jing, LI Dong

(Software Quality Engineering Research Center, The 5th Electronics Research Institute, Ministry of Industry and Information Technology, Guangzhou 510610, China)

【Abstract】This paper proposes a software reliability model to resolve the software reliability growth problem in the distributed development environment. It does experiment on practical software reliability data set to demonstrate the model through the use of adaptive measures and AIC criterion. Experiments show that this model can capture a large number of reliability growth curves, has strong flexibility and adaptability, and a distributed development environment described by this model can be more realistic.

【Key words】distributed system; software test; software reliability; reliability model

DOI: 10.3969/j.issn.1000-3428.2012.03.018

1 概述

分布式系统在客户端-服务器架构环境中有着广泛的应用和发展。在分布式系统中，客户端软件发送请求以获得服务或数据，处于另一端的服务器在接收到请求后决定是否进行响应，这样的模式构成了当今网络的基础。

在分布式系统中，客户端软件程序的设计开发需要结合传统的程序设计原理、概念和方法，以及面向对象的概念和经典的软件工程方法。在其测试阶段，对软件故障的发生和移除现象进行建模和分析日趋重要，当前已有大量研究工作对此展开论述，这些研究的共同目的是开发出一个分析故障移除现象的模型，用于计算软件的故障移除数量，残余故障数量及评价软件可靠性。这些数据对软件系统开发和设计阶段有着积极的意义。

文献[1-6]提出了多个用于监控的软件故障移除流程，以及衡量和预测软件系统可靠性的模型。在该模型中，观察显示测试时间和故障移除数量之间的关系表示为指数型、*S* 型或者两者的结合体，它在不需要知道任何被测试软件环境特性信息的情况下就可以应用，属于黑盒模型。

在白盒测试中，测试者需要了解软件开发所用到的技术，因而开发一个软件可靠性模型来精确解释软件开发和测试过程中用到的技术，同时确切地考虑软件故障的严重程度就显得十分重要。文献[7-8]最早采用该建模方法，用来收集软件成长曲线中由应用环境带来的变化数据，该模型可以削弱指数曲线或者 S 型曲线的增长。因此，这样的模型更适用于面向对象和分布式开发环境。

软件模块的重用可以在很大程度上扩大商业利益，提高产品质量和开发效率并降低总成本。在理想情况下，用于重用的软件模块必须通过完整的验证并且没有缺陷。但是在现实情况下，正规验证不会定期进行，bug 也不一定可以重现。因此，在软件重用过程中必然要先发现 bug 然后去消除，这样软件的质量才能逐步提升。根据惠普公司的一项研究，重用软件的缺陷率在 0.9/KLOC，而全新开发的软件缺陷率为 4.1/KLOC。同时可重用的软件组建在其软件部署流程中会花费更少时间来进行计划、建模、编写文档、编码和组织数据等工作。因此，生产效率的提高不仅因为编码时间的减少，还得益于测试和编写文档的时间的减少[9-10]。

本文提出了一个适用于分布式系统的可靠性模型，并定义其软件可靠性策略，探讨了进行参数估计的方法以及用于验证和评价该模型的标准条件，且对使用该模型所得到的可靠性数据进行了分析和比较。

2 可靠性建模

在分布式开发环境下，软件模块可以在不同的地方开发部署，同时也可以有很多模块被其他软件所重用。

文献[11]提出一个适用于分布式开发环境的基于非齐次泊松过程(Non-homogeneous Poisson Process, NHPP)的软件可靠性模型，该模型组合了文献[12]之前提出的延迟型 S 形模型和指数级模型。该模型基于 2 条假设：(1)软件系统由 *n* 个重用模块和 *m* 个新开发模块构成；(2)当被测软件系统包含一定数量的重用组件时，随着测试工作的推进，该系统所检测

基金项目：“核高基”重大专项“通用基础软件测试评估”(2009ZX 01045-004-002)

作者简介：杨春晖(1965－)，女，研究员级高级工程师，主研方向：软件测试，软件可靠性技术；熊 婧，助理工程师、硕士研究生；李 冬，高级工程师、博士

收稿日期：2011-08-04 **E-mail：**jing.xiong8508@gmail.com

出的故障数目的累计值变化符合指数型曲线的变化趋势；而当系统新开发组件被使用后，所检测出的故障数目的累计值变化趋势类似于 S 型曲线。

Yamaa 提出的模型表示为：

$$H(t)=a[\sum_{i=1}^{n}p_i\left(1-\exp(-b_it)\right)+\sum_{j=1}^{m}p_{n+j}\left\{1-\left(1+b_{n+j}t\right)\exp(-b_{n+j}t)\right\}]$$

其中，a 表示软件固有故障的预期数目；b_i 代表软件系统第 i 个模块中每一个固有故障所引发系统失效的概率；权重参数 p_i 表示系统第 i 个模块在整个系统测试负载中所占的比例，满足 $p_i>0$ 和 $\sum p_i=1$，且取值为 $p_i=1,2,\cdots,n+m$。

此模型只适用于描述纯指数型增长曲线和 S 型增长曲线。本文提出一种依赖于环境的软件可靠性模型，该模型在故障移除阶段合并了一个逻辑学习函数，通过捕获范围更广的可靠性增长曲线估计测试权重参数的值，能够适用于从纯指数型曲线到 S 型曲线之间更多类别的曲线。

2.1 模型假设

本文所提出的模型基于以下假设：

(1)软件故障移除现象遵循均值函数为 $m(t)$的非齐次泊松过程。

(2)软件系统运行期间容易发生由遗留故障导致的失效。

(3)软件系统由有限数量的新开发模块和重用模块组成。

(4)软件系统中重用模块和新开发模块中发生的故障比例大约为 1:4。

(5)软件系统可靠性在所有重用模块中的增长保持一致，在新开发模块中不一致。

(6)软件发生失效时，系统在一定时间的延迟后移除导致该失效的故障。观察到系统发生失效时刻与随后故障移除时刻之间的延迟长短表示该故障的严重程度。

(7)软件系统的故障移除过程是完美的。

(8)软件系统重用模块的故障移除率保持恒定。

(9)软件系统新开发模块的故障移除率可以表示为一个逻辑函数。

(10)在时间间隔$(t, t+\Delta t)$内，所预期移除的故障数目与剩余故障数目存在一定的比例。

2.2 模型定义

本文所提模型中将使用的符号如表 1 所示。

表 1 模型符号

符号	含义
a	软件系统所有故障($\sum a_i+\sum a_j=a$)
$a_i(=ah_i)$	软件系统中 i 类型($i=1,2,\cdots,m$)故障的总数
$a_j(=ah_j)$	j 类型故障的总数($j=1,2,\cdots,n$)
$h_i(h_j)$	i 类型故障与 j 类型故障数目的比例($0<h_i\leqslant 0.2$, $0<h_j\leqslant 0.8$, $\sum h_i+\sum h_j=1$)
b_i	i 类型故障导致的系统失效率/i 类型故障的故障移除率/i 类型故障的故障隔离率
b_j	j 类型故障导致的系统失效率/j 类型故障的故障隔离率
$b_j(t)$	逻辑学习函数，例如 t 时刻，j 类型故障的故障移除率
$m_{jf}(t)$	t 时刻由 j 类型故障导致的平均软件失效次数
$m_{js}(t)$	t 时刻与 j 类型故障无关的平均故障数
$m_{jr}(t)$	t 时刻所移除的 j 类型平均故障数
β	逻辑学习函数中的恒定参数

下面分别对重用模块和新开发模块的故障移除现象进行建模和分析：

(1)重用组件的故障移除过程称为简单的故障移除过程，第 i 个重用模块的故障移除过程定义为：

$$\mathrm{d}(m_{ir}(t))/\mathrm{d}t=b_i\left(a_i-m_{ir}(t)\right) \tag{1}$$

式(1)中的一步法描述了系统失效观察、故障隔离以及故障移除的过程。在条件 $m_{ir}(t=0)=0$ 下，由式(1)可得到下式：

$$m_{ir}(t)=a_i\left(1-\exp(-b_it)\right) \tag{2}$$

(2)新开发组件的故障移除过程称为复杂的故障移除过程，第 j 个新开发模块的故障移除过程定义为：

$$\mathrm{d}(m_{jf}(t))/\mathrm{d}t=b_j\left(a_j-m_{jf}(t)\right) \tag{3}$$

$$\mathrm{d}(m_{js}(t))/\mathrm{d}t=b_j\left(m_{jf}(t)-m_{js}(t)\right) \tag{4}$$

$$\mathrm{d}(m_{jr}(t))/\mathrm{d}t=b_j(t)\left(m_{js}(t)-m_{jr}(t)\right) \tag{5}$$

其中，$b_j(t)=b_j/(1+\beta\exp(-b_jt))$。

式(3)表示观察所得到的软件系统失效情况，式(4)描述软件系统故障隔离过程，式(5)则表示软件系统的故障移除过程。在 $m_{jf}(t=0)=0$、$m_{js}(t=0)=0$、$m_{jr}(t=0)=0$ 下，可推导得：

$$m_{jr}(t)=a_j(1-(1+b_jt+b_j^2t^2/2)\exp(-b_jt))/(1+\beta\exp(-b_jt)) \tag{6}$$

2.3 故障移除

在本文所提出的模型中，NHPP 均值函数是由式(2)和式(6)中所定义的 NHPP 均值函数叠加而成的。因此，叠加后的 NHPP 均值函数定义为：

$$m_r(t)=\sum_{i=1}^{m}m_{ir}(t)+\sum_{j=m+1}^{n}m_{jr}(t)=\sum_{i=1}^{m}a_i\left(1-\exp(-b_it)\right)+\sum_{j=m+1}^{n}a_j(1-(1+b_jt+b_j^2t^2/2)\exp(-b_jt))/(1+\beta\exp(-b_jt)) \tag{7}$$

设某个软件系统由 2 个重用模块和 2 个新开发模块组成，则式(7)所提出的模型可表示为：

$$m_r(t)=\sum_{i=1}^{2}a_i\left(1-\exp(-b_it)\right)+\sum_{j=3}^{4}a_j(1-(1+b_jt+\frac{b_j^2t^2}{2})\exp(-b_jt))/(1+\beta\exp(-b_jt)) \tag{8}$$

其中，$a_1=ah_1$；$a_2=ah_2=a(0.2-h_1)$；$a_3=ah_3$；$a_4=ah_4(0.8-h_3)$；$b_1=b_2$；$b_3=b_4$；$a_1+a_2+a_3+a_4=a$。

3 参数估计方法

极大似然估计(Maximum Likelihood Estimation method, MLE)常用于估计式(8)中的未知参数。二元组$(t_i,x_i)(i=1,2,\cdots,k)$表示所使用的数据集，$x_i$ 表示时刻 $t_i(0<t_1<t_2<\cdots<t_k)$所移除的累计故障数目，$t_i$ 表示系统移除 x_i 个故障数目花费的累计时间。因此，包含未知参数的似然函数 L 表示为：

$$L\left(parameters\mid(t_i,x_i)\right)=\prod_{i=1}^{k}\frac{\left[m(t_i)-m(t_{i-1})\right]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}\exp\left(-\left(m(t_i)-m(t_{i-1})\right)\right)$$

对 L 取自然对数后得到 $\ln L$ 的表达式如下：

$$\ln L=\sum_{i=1}^{k}(x_i-x_{i-1})\ln\left[m(t_i)-m(t_{i-1})\right]-\left\{m(t_i)-m(t_{i-1})\right\}-\sum_{i=1}^{k}\ln\left[(x_i-x_{i-1})!\right]$$

在满足 $a>0$，$0<h_i\leqslant 0.2$，$0<b_i<1$，$0<h_j\leqslant 0.8$，$0<b_j<1$，$\beta\geqslant 0$ 的情况下，求取似然函数 L 的最大值，从而可以得到上述参数的最大似然估计。

4 模型的验证与评价方法

为了验证式(8)给出的模型的有效性，用以描述软件可靠性增长试验，在一个真实的软件开发项目中所得来的实际可靠性数据上进行了测试。该数据集是从对一个使用 PL/1 语言

开发、逻辑大小为 1 317 KB 的数据库应用程序进行为期19周的测试过程中收集而来的，共有328个故障被移除。

一个软件可靠性模型的性能是根据它接收过去软件故障数据的适应能力，以及根据过去和现在的数据行为预测未来行为的能力即预测效度来评价的。

(1)适应性度量。该度量表示实际数据与模型估计值之间的差异，表示为：

$$SSE=\sum_{i=1}^{k}\left(\hat{m}(t_i)-x_i\right)^2$$

观察次数 $\hat{m}(t_i)$ 表示时刻为 t_i 时，根据式(8)中的均值函数计算得来的累计故障数目，x_i 表示时刻为 t_i 时所移除的故障总数，*SSE* 取值越小则说明适应性越强。

(2)AIC 准则。AIC 准则最初是作为一个软件可靠性模型选取工具而提出来的，根据 AIC 计算公式，即：

$$AIC=-2\times\lg(\max\ \text{of}\ \text{Likelihood}\ \text{function})+2\times N$$

其中，N 表示模型中使用的参数个数。

较小的 *AIC* 值说明模型的适应性和预测能力越高。相当于使用 *SSE* 和 *AIC* 的度量值来评价模型的性能。在相同的数据集上，小于其他模型的 *SSE* 和 *AIC* 值，则说明该模型的适应性和预测能力要优于其他模型。

5 结果对比分析

根据观察，应用MLE方法时该模型中均衡参数(h_1, h_2, h_3, h_4)的值分别为(0.168 6, 0.031 4, 0.545 5, 0.254 5)。而 Yamada 则假设这些值为(0.05, 0.05, 0.45, 0.45)。

表2给出了适应性度量和 *AIC* 计算的结果。结果显示在测试过程中，测试小组的技能都得到了提高，而根据适度矩阵，该模型的结果是最好的。

表2 模型验证及评价结果对比

模型对比	参数估计				验证及评价	
	a	$b_{1,2}$	$b_{3,4}$	β	*SSE*	*AIC*
Yamada 模型	378.12	0.465 4	0.178 8	—	2 374.73	213.67
本文模型	364.73	0.485 9	0.284 6	2.004 4	1 173.88	212.56

表3给出了模块故障内容的对比。表1和表2显示了实际累积的故障和不同模型适度的对比。可以明显地看出，推荐的模型比 Yamada 模型更加符合实际情况。

表3 不同模块故障内容对比

模型对比	重用模块		新开发模块	
	a_1	a_2	a_3	a_4
Yamada 模型	18.91	18.91	170.15	170.15
本文模型	61.50	11.45	198.97	92.81

6 结束语

在分布式系统的开发过程中，客户端-服务器架构被视为一个平台。在20世纪70年代和20世纪80年代，软件系统向集中化方式的转变产生了单一的巨大系统；而在 20 世纪80年代末向非集中式的方向发展。多数厂商都从开发纯粹的系统软件产品转向开发活跃的异构系统。

本文提出了基于 NHPP 的模型，用于评估分布式开发环境中软件可靠性的生长现象。将模型应用到实际的软件开发项目中得到的数据来对该模型进行证实和评估，结果表明：从纯粹的指数型到高度的 S 型，该模型能捕获大量的可靠性成长曲线，并且具备较强的适用性和灵活性。因此，使用该模型所描述的分布式软件开发环境更加符合实际情况。

参考文献

[1] Musa J D, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Applications[M]. New York, USA: McGraw-Hill, 1987.

[2] Xie Min. Software Reliability Modelling[M]. New York, USA: World Scientific, 1991.

[3] Khoshogoftaar T M, Woodcock T G. Software Reliability Model Selection: A Case Study[C]//Proc. of International Symposium on Software Reliability Engineering. Austin, USA: [s. n.], 1991.

[4] Lyu M. Handbook of Software Reliability Engineering[M]. New York, USA: McGraw-Hill, 1996.

[5] Musa J D. Software Reliability Engineering[M]. New York, USA: McGraw-Hill, 1999.

[6] 艾 骏, 陆民燕, 阮 镰. 面向软件可靠性测试数据生成的剖面构造技术[J]. 计算机工程, 2006, 32(22): 7-9, 45.

[7] Pham H. Software Reliability[M]. [S. l.]: Springer-Verlag, 2000.

[8] Kapur P K, Bardhan A K, Omar S. Why Software Reliability Growth Modelling Should Define Errors of Different Severity[J]. Quality Control and Applied Statistics, 2004, 49(6): 699-702.

[9] Kapur P K, Omar S, Yadavalli V S S. A Software Fault Classification Model[J]. South African Computer Journal, 2004, 33(12): 1-9.

[10] Pressman R S. Software Engineering: A Practitioner's Approach[M]. 5th ed. New York, USA: McGraw-Hill, 2001.

[11] Yamada S, Ohba M, Osaki S. S-shaped Reliability Growth Modelling for Software Error Detection[J]. IEEE Transactions on Reliability, 1983, 32(6): 475-478.

[12] Yamada S, Tamura Y, Kimura M. A Software Reliability Growth Model for a Distributed Development Environment[J]. Electronics and Communications in Japan, 2000, 83(12): 1-8.

编辑 任吉慧

(上接第50页)

参考文献

[1] Wang Ping, Chao Kuo-Ming, Lo Chi-Chun. On Optimal Decision for QoS-aware Composite Service Selection[J]. Expert Systems with Applications, 2010, 37(1): 440-449.

[2] Tsesmetzis D, Roussaki I. Modeling and Simulation of QoS-aware Web Service Selection for Provider Profit Maximization[J]. Simulation, 2007, 83(1): 93-106.

[3] Zeng Liangzhao, Benatallah B. QoS-aware Middleware for Web Services Composition[J]. IEEE Transactions on Software Engineering, 2004, 30(5): 311-327.

[4] Huang A F M, Lan Ci Wei , Yang S J H. An Optimal QoS-based Web Service Selection Scheme[J]. Information Sciences, 2009, 179(19): 3309-3322.

[5] Liu Feng, Lei Zhenming. Research on User-aware QoS Based Web Services Composition[J]. The Journal of China Universities of Posts and Telecommunications, 2009, 16(5): 125-130.

[6] Hwang San-Yih, Wang Haojun, Tang Jian. A Probabilistic Approach to Modeling and Estimating the QoS of Web-services-based Workflows[J]. Information Sciences, 2007, 177(23): 5484-5503.

[7] 代 钮, 杨 雷, 张 斌, 等. 支持组合服务选取的模型及优化求解[J]. 计算机学报, 2006, 29(7): 1167-1178.

[8] 王成良, 冯 欣. 基于目标递进的 Web 服务组合方法[J]. 计算机工程, 2011, 37(6): 52-54.

编辑 索书志