P

·计算机工程与应用·

软件可靠性模型现状与研究

耿 技[1]，聂 鹏[1,2]，秦志光[1]

(1. 电子科技大学计算机科学与工程学院 成都 611731; 2. 江西财经大学计算机实践教学中心 南昌 330013)

【摘要】对近几十年中用于正确评估预测软件系统可靠性的主要模型进行了回顾性研究。对基于模型采用的数学工具，将文献中的模型分为解析性模型和启发性模型。再对于模型假设以及模型主要思想进行了研究和分析。最后对软件系统可靠性模型研究趋势和存在的问题进行了展望。

关 键 词 解析性模型; 启发性模型; 可靠性模型; 软件可靠性
中图分类号 TP311 文献标志码 A doi:10.3969/j.issn.1001-0548.2013.04.007

Status and Research of Software Reliability Models

GENG Ji[1], NIE Peng[1,2], and QIN Zhi-guang[1]

(1. School of Computer Science & Engineering, University of Electronic Science and Technology of China Chengdu 611731;
2. Computer Center of Practice Teaching, Jiangxi University of Finance and Economics Nanchang 330013)

Abstract An overview study was made on prime software reliability models proposed by the researchers in the last decades for accurate reliability analysis and forecasting. Firstly, the major models are classified into the analytic models and the heuristic models based on their mathematical theories. Secondly, studies and analyses on the model assumptions and main ideas are presented in this paper. Finally, the current trends and existing problems in the software reliability models are addressed.

Key words analytic model; heuristic model; reliability model; software reliability

在近几十年中，计算机软件从代码体积和复杂度两个方面呈现出指数性增长。软件指数性增长趋势更放大了低可靠性软件所产生的破坏范围，因此软件自身的可靠性成为不可忽视的关键问题。

IEEE把软件可靠性定义为在规定条件下，在规定时间内，软件不发生失效的概率[1]。软件可靠性模型(software reliability model，SRM)是软件可靠性研究中备受关注、成果最多、最活跃的一个领域。从1972年J-M软件可靠性模型[2]发表到今天，已公开发表的模型有几百种。软件可靠性模型旨在根据软件失效数据，通过建模给出软件的可靠性估计值或预测值。它不仅是软件可靠性预计、分配、分析与评价的最强有力的工具，而且为改善软件质量提供了指南。

软件可靠性模型从建模方法上可以分为软件可靠性解析模型和软件可靠性启发模型两大类。软件可靠性解析模型主要通过对软件失效数据行为进行假设，并在该假设的基础上依靠数学解析方法对软件可靠性建模；软件可靠性启发模型仅依赖软件历史失效数据，首先建立可靠性模型原型，然后让模型原型对软件历史失效数据进行学习，达到自我优化的目的，最终逼近实际的软件可靠性。

1 软件可靠性解析模型

软件可靠性解析模型主要通过对软件失效数据行为进行假设，并在该假设的基础上依靠数学解析方法对软件可靠性进行建模。该类模型可分为指数模型、对数模型、Littlewood-Verrall模型、数据域模型、Markov链模型、随机Petri网模型等。

收稿日期：2011－09－15；修回日期：2011－11－22
基金项目：国家自然科学基金(60973118)
作者简介：耿技(1963－)，男，博士生，教授，主要从事系统软件、软件确保、软件可靠性方面的研究.

1.1 指数模型

1.1.1 J-M模型

J-M 软 件 可 靠 性 模 型 于 1972 年 由 Jelinski 和 Moranda创建[2]，属于二项分布有限错误模型。其基本假设如下：1) 测试未运行时软件失效为0；当测试进行时，软件错误将被检出，其失效率函数与软件当前的残留故障数成正比；同时，软件中存在的总故障数是固定的；2) 失效率在每个失效间隔内是常数，其数值正比于残留的错误数；3) 软件错误引发的故障是相互独立的；4) 每次只修正一个错误，且当软件故障出现时，引发故障的错误将被立即排除，并不会引入新的错误。

J-M 模 型 中 软 件 第 i 次 失 效 强 度 函 数 为 $\lambda_i = \Phi(N-i+1)$，其中N为软件中错误总数，Φ 为失效率。J-M模型软件可靠性函数为：

$$R(x_i) = \mathrm{e}^{-\lambda_i x_i} \tag{1}$$

J-M模型以一种较为简单的方式，将软件故障视为测试时间的函数，主要缺点在于假设条件过于理想，实际情况中很难满足。

1.1.2 G-O模型

Goel-Okumoto软件可靠性模型[3](G-O模型)于1979年由Goel和Okumoto提出，属于NHPP[4-5]有限错误模型，其基本假设如下：1) 测试未运行时的软件失效为0；当测试进行时，软件失效服从均值为 $m(t)$ 的泊松分布；2) 当 $\Delta t \to 0$ 时，测试时间 $(t, t+\Delta t]$ 内产生的失效与软件残留错误成正比；3) 对于任一组有限时间点，在对应时间段分别发生的失效次数相互独立；4) 每次只修正一个错误，当软件故障出现时，引发故障的错误被立即排除，并不会引入新的错误。

G-O模型在测试区间 $[0,t]$ 内的累计失效数期望函数为 $m(t) = a(1-e^{-bt})$，t 为软件累计测试时间。可靠性函数为：

$$R(s \mid t) = \mathrm{e}^{a(\mathrm{e}^{-b(t+s)} - \mathrm{e}^{-bt})} = \mathrm{e}^{m(s)\mathrm{e}^{-bt}} \tag{2}$$

1.1.3 Musa基本执行时间模型

Musa基本执行时间软件可靠性模型[6](Musa模型)中，软件测试时间使用了更为精确的CPU占用时间作为度量基础，并给出了CPU时间与日历时间的转换关系，但软件由于运行环境的差异导致CPU执行时间可能大不相同。

1.1.4 超指数增长模型

超指数增长模型[7]对经典指数模型进行了扩展，属于NHPP有限错误模型。由于编程人员的差异、新旧代码的差异、实现语言的差异等因素导致了软件不同部分的失效率各有不同，因此软件的不同部分将分配不同的失效率[8]。超指数增长模型可用于拥有新模块和重用模块的复杂系统中。超指数增长模型在测试区间 $[0,t]$ 内的累计失效数期望函数为：

$$m(t) = \sum_{i=1}^{n} a_i (1 - \mathrm{e}^{-b_i t}) \tag{3}$$

式中，n 表示具有相似特征模块构成的簇数量；a_i 表示在第 i 个簇中存在的错误总数；b_i 表示在第 i 个簇的失效率。该模型在应用中如何确定大型复杂系统中模块簇的划分是该模型的关键问题之一。

1.1.5 S-Shape模型

S-Shape模型主要分为Yamada Delayed S-Shaped模型[9]和Inflected S-Shaped模型[10]。S-Shape模型使用Gamma分布取代了G-O模型的二项分布，属于NHPP有限错误模型。S-Shape模型认为错误发现曲线应该体现出两个特征：1) 软件测试者对测试软件的了解程度；2) 软件残余缺陷随测试过程的进行，其发现难度变得越来越大。

Yamada Delayed S-Shaped模型认为软件失效在测试开始时增大，在测试结束时减小。在该模型中，软件失效数据随测试时间呈S曲线变化，体现出由于测试初期测试者对测试对象的不熟悉以及测试后期失效数据难于发现导致失效发现率下降这一现象。

Yamada Delayed S-Shaped模型在测试区间 $[0,t]$ 内的累计失效数期望函数为：

$$m(t) = a(1 - (1+bt)\mathrm{e}^{-bt}) \tag{4}$$

式中，a 表示最终被检测出的失效总数的期望值；b 表示失效率。

Inflection S-Shaped模型认为错误发现率在整个测试周期中呈现递增的趋势。Inflection S-Shaped模型在测试区间 $[0,t]$ 内的累计失效数期望函数为：

$$m(t) = \frac{a(1-\mathrm{e}^{-bt})}{1+c\mathrm{e}^{-bt}} \qquad c = \frac{1-r}{r} \tag{5}$$

式中，a 表示最终被检测出的失效总数的期望值；b 表示失效率；r 表示可以检测的失效数占软件系统总失效数的比例。

1.2 对数模型

1.2.1 Geometric模型

Geometric模型[11-12]属于对数无限错误模型，其基本假设如下：1) 失效发现率随软件检测过程递减；2) 系统中错误是无限的。

该模型认为较早发现的错误对失效发现率的影响大于较晚发现的错误，其错误发现间隔时间为指数分布，错误发现间隔时间密度为 $f(X_i)=D\phi^{i-1}\mathrm{e}^{-D\phi^{i-1}X_i}$，其中 D 和 ϕ 为常量参数，X_i 为第 i 个失效数据发生时的观测数据。Geometric模型中从第 $(i-1)$ 个错误开始，发现第 i 个错误的期望时间为 $E(X_i)=\frac{1}{D\phi^{i-1}}$。Geometric模型在测试区间 $[0,t]$ 内的累计失效数期望函数为：

$$m(t)=\frac{1}{\beta}\ln(D\beta\mathrm{e}^{\beta}t+1) \tag{6}$$

式中，$\beta=-\ln(\phi)$ 且 $0<\phi<1$。

1.2.2 Musa-Okumoto对数泊松模型

Musa-Okumoto对数泊松模型[13]属于对数无限错误模型。该模型构建于指数递减的失效强度函数之上，其失效强度伴随失效期望递减成指数递减，即 $\lambda(t)=\lambda_0\mathrm{e}^{-bm(t)}$，$b>0$ 为失效率衰减参数，$\lambda_0>0$ 为初始失效率。

1.3 Littlewood-Verrall模型

Littlewood-Verrall可靠性模型(L-V模型)[14]考虑了发现缺陷不被完全剔除的情况。该模型基本假设如下：1) 相邻错误间隔时间 X_i 为相互独立指数随机变量 ξ_i，其中 $i=1,2,\cdots,n$；2) ξ_i 构成独立随机变量序列，并服从参数为 α 和 $\psi(i)$ 的Gamma分布，$\psi(i)$ 为增函数，用于描述程序员的素质以及软件开发难度；3) 软件使用情况近似于设计预期。

增函数 $\psi(i)$ 综合了程序员素质 β_0 以及软件开发难度 β_1 两个因素，在L-V模型中 $\psi(i)$ 的线性表达式和二次表达式分别为 $\psi(i)=\beta_0+\beta_1 i$ 和 $\psi(i)=\beta_0+\beta_1 i^2$，其中 β_0、$\beta_1>0$。

L-V模型的累计失效数期望函数为：

$$m(t)=\frac{1}{\beta_1}(-\beta_0+\sqrt{\beta_0^2+2\beta_1 t}) \tag{7}$$

L-V模型与前面讨论的模型相比，最主要的不同在于前面的模型仅考虑了错误的发生对软件可靠性的影响，而没有考虑软件稳定运行这一现象对软件可靠性的影响。

1.4 数据域模型

Nelson模型是数据域软件可靠性模型的代表，也是最重要的软件可靠性模型之一。该模型最早于1973年由Nelson提出[15]，并于1978年得以完善[16]。Nelson模型中，软件的可靠性通过对软件运行的输入数据进行测量，这些输入数据从集合 $E=(E_i:i=1,2,\cdots,N)$ 中随机选取。随机选取的 n 个输入数据概率分布为 $(P_i:i=1,2,\cdots,N)$。

Nelson模型基本假设如下：1) 程序被认为是集合 E 上的一个可计算函数 F 的一个规范，一个输入数据 E_i 对应一个程序执行回合并产生一个输出 $F(E_i)$；2) 由于程序包含缺陷，程序实际确定函数 F'，该函数不同于希望函数 F；3) 对于某些 E_i，程序实际输出 $F'(E_i)$ 在希望输出 $F(E_i)$ 的容许范围之内，即 $|F'(E_i)-F(E_i)|\leqslant\Delta_i$；但对另一些 E_j，程序实际输出 $F'(E_j)$ 超出容许范围，即 $|F'(E_j)-F(E_j)|>\Delta_j$，这时认为程序发生一次失效；4) 测试过程中不剔除程序缺陷。

若 n_e 为导致软件产生故障的输入数据数，则软件可靠性可以表示为 $R_1=1-\frac{n_e}{n}$。另一个软件可靠性计算公式 $R_2=1-\sum_{j=1}^{N}(\frac{f_j}{n_j})p(E_j)$ 由文献[17]提出，其中 n_j 为输入子域 E_j 中元素的数量，f_j 为 n_j 次运行后软件出现故障的次数。

Nelson模型存在的问题主要表现在输入数据集较大和输入数据的随机选取具有盲目性和局限性。

1.5 Markov链模型

基于Markov链的软件可靠性模型主要用于评估预测基于构件的软件系统。构件是指封装了数据和功能的，在运行时能够通过参数进行配置的模块。通常构件由第三方开发，具有清晰的接口描述。随着软构件技术的快速发展，聚集软构件设计大型复杂软件系统的软件开发方法日趋成熟，基于构件的可靠性模型研究也得到越来越多的关注。

1.5.1 Cheung模型

Cheung模型由文献[18]提出，属于基于Markov链的软件可靠性模型。Cheung模型将软件的控制结构转化成有向图 G，并规定图中每一个节点 N_i 表示一个构件，构件 N_i 到 N_j 的转移用一个矢量边 (N_i,N_j) 表示。定义 R_i 为 N_i 的可靠性，P_{ij} 为 (N_i,N_j)

的转移概率。根据有向图G，构造出构件转移矩阵$\boldsymbol{M}$，其中$M(i,j)=R_iP_{ij}\,M(i，j)$，表示从N_i成功到达N_j的可能性。

Cheung 模型基本假设如下：1) 构件间的可靠性是相互独立的；2) 构件间的控制转移是马尔科夫过程；3) 构件间连接逻辑完全可靠。

Cheung模型中，具有n个构件的软件系统可靠性表示为$R=\boldsymbol{S}(1,n)\times R_n$，其中$R_n$为第$n$个构件的个体可靠性，$\boldsymbol{S}$为$n\times n$矩阵，具体为：

$$\boldsymbol{S}=\boldsymbol{I}+\boldsymbol{Q}+\boldsymbol{Q}^2+\boldsymbol{Q}^3+\cdots=\sum_{k=0}^{\infty}\boldsymbol{Q}^k \tag{8}$$

式中，$\boldsymbol{I}$为$n\times n$单位矩阵；$\boldsymbol{Q}$为转移概率矩阵$\boldsymbol{P}$的变形。Cheung模型只能处理单输入/单输出系统，没有考虑连接件的可靠性，而且把构件的可靠性假设为固定不变的参数。

1.5.2 Krishnamurthy模型

Krishnamurthy 模型由文献[19]提出，遵守Cheung模型的假设条件，属于Markov链软件可靠性模型。该模型将软件中的构件视为独立节点，将任意一个测试用例运行中经过的节点序列视为路径。基于这一基本概念，该模型认为基于构件的软件系统的可靠性可以由路径的可靠性予以描述。

1.5.3 Yacoub模型

Yacoub 模型由文献[20]提出，属于 Markov 链软件可靠性模型。该模型用构件依赖图来描述构件间的组装交互关系，构件依赖图是一个有向图，用于描述构件的可靠性、连接与接口的可靠性、构件间控制的转换及转换的概率等方面内容。该模型认为构件间连接逻辑并非完全可靠，并引入RT_{ij}进行量化，但模型中并未对此做深入分析和讨论，而仅作为固定不变的参数值加以使用。

1.5.4 毛晓光通用模型

毛晓光通用模型[21]认为软件可靠性的计算是软件中所有运行路径出现频度与该路经可靠性乘积的累加$R=\sum_{P_i\in\varphi}R_{P_i}\times F_{P_i}\Big/\sum_{P_i\in\varphi}F_{P_i}$，$R$为可靠性，$\varphi$为所有运行路径的集合，$R_{P_i}$和$F_{P_i}$分别为路径$P_i$的可靠性和出现频度。该模型没有对连接件的可靠性进行分析，并且没有涉及敏感度计算，不利于对软件早期开发的指导。

1.5.5 Wang模型

Wang模型由文献[22-23]提出，该模型对Cheung模型进行了改进，允许软件具有并发、异构、多输入/输出特性。该模型将输入和输出分别定义为$I=\{S_{i1},S_{i2},\cdots,S_{im}\}$和$F=\{S_{f1},S_{f2},\cdots,S_{fn}\}$，同时增加了超级初态$S^I$和超级终态$S^F$，$S^I$与$S^F$的可靠性均为1。Wang模型对不同结构的软件体系结构分别进行可靠性分析，在状态转移概率中考虑了连接件的可靠性，但这样处理的连接件的可靠性并不符合实际。

1.6 随机Petri网模型

随机Petri网作为软件可靠性建模的一种工具[24]能较全面地描述系统的动态变化行为，当系统发生变化时，只要增加相应位置中的标记数即可，可以大大减少工作量。该模型将具体系统转化为随机Petri网模型；并构造出与该随机Petri网模型同构的Markov链；最后基于Markov链状态概率进行系统可靠性分析。状态S代表可能发生的局部故障状态；变迁T代表使系统状态变化的事件；弧R表示S和T间的关系。随机Petri网的软件可靠性模型示意图如图1所示。

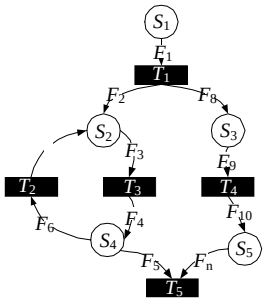

图1 随机Petri网的软件可靠性模型示意图

文献[25]针对嵌入式系统进行了系统可靠性建模。文献[26]针对大规模软件利用Petri网进行了建模。文献[27]提出采用构建Petri子网和Petri网约简两种方法对Petri网软件可靠性模型中存在的状态爆炸问题进行改进。文章[28]提出Reliability Petri Net (RPN)，该模型在传统弧上附加了故障状态和失效事件关系产生的可靠性指标，并给出了顺序运算、分支运算、并行运算和复合运算4种可靠性计算规则。文献[29-31]分别对数据存储系统、内燃机控制软件、基于WEB的服务应用使用Petri网进行了可靠性建模和分析。

2 软件可靠性启发模型

软件可靠性启发模型不同于解析模型，该模型

利用软件历史失效数据对自身进行训练、更新，逼近实际可靠性。启发模型可分为基于神经网络的软件可靠性模型和基于遗传编程的软件可靠性模型。

2.1 神经网络模型

文献[32]将神经网络模型应用于软件可靠性建模，突出优点在于无需对软件系统的失效进行假设约束。文献[33]实验性证明了增加隐藏层可以有效地提高神经网络模型的性能。文献[34]使用小波神经网络对软件可靠性进行建模，并验证该模型的有效性。文献[35]使用构件故障进行建模分析，证明该方法可以节约可靠性评估/预测时间。文献[36]使用模糊小波神经网络对软件可靠性进行建模，实验证明该模型构建方法简单。文献[37]提出采用自组织算法，优化设计隐含层神经网络，使用可变学习速率提高了标准BP神经网络模型的预测精确度。

2.2 遗传编程

遗传编程(genetic programming，GP)[38]是最近几年提出并得以发展的软件可靠性建模方法。

文献[38]提出了基于时间GP模型和测试覆盖率GP模型，并将GP模型与传统解析性软件可靠性模型和神经网络软件可靠性模型进行了比较。GP模型采用遗传算法的基本思想，采用树形结构表示失效数据表达式，叶节点是失效的原始变量，中间节点为失效数据函数，根节点是最终输出的失效数据曲线表达式。通过使用交叉、变异等遗传操作和选择操作动态地改变这些树结构，并迭代演化直至找到失效数据曲线的优化表达式。

文献[39]使用Boosting算法对GP软件可靠性模型进行改进，新的模型GP-Boosting可以减少有差异的函数集合，降低GP模型的配置难度。文献[40]进一步证明了GP模型较神经网络模型与解析性模型具有更好的可靠性评估能力。文献[41]中对GP模型、GPB模型、神经网络模型和传统解析性模型进行了详细的比较和分析。文献[42-43]对GP模型的适应度函数进行改进，提出了基于应用可适性的GP模型。

3 结 论

现有的软件可靠性模型从建立模型的基本方法上可以分为解析模型和启发模型。建立在概率论和数理统计基础上的解析模型最早引入并发展了近40年。由于软件日趋复杂多样，解析模型的假设条件往往不能完全满足，导致模型的适用性下降。因此，软件可靠性模型假设、优化、优选和适用范围等问题有待进一步研究。

随着软件可靠性理论发展，新的数学工具，如GP、神经网络等技术的引入，形成了软件可靠性启发模型。软件可靠性模型研究已经认识到软件可靠性评估预测单纯依赖失效数据是不够的，对于复杂的软件系统而言，将其结构特征引入到软件可靠性的研究工作中是非常必要的。

参 考 文 献

[1] IEEE. IEEE recommended practice on software reliability [EB/OL]. [2011-09-03]. http://www.bzfxw.com/soft/gwbz/IEEE-Standards/14485395.html.

[2] JELINSKI Z, MORANDA P B. Software reliability research[J]. Statistical Computer Performance Evaluation, 1972, 22(1): 465-484.

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

[4] 谢景燕, 安金霞, 朱纪洪. 考虑不完美排错情况的NHPP类软件可靠性增长模型[J]. 软件学报, 2010, 21(5): 942-949.
XIE Jing-yan, AN Jin-xia, ZHU Ji-hong. NHPP software reliability growth model considering imperfect debugging[J]. Journal of Software, 2010, 21(5): 942-949.

[5] 李海峰, 李秋英, 陆民燕. 基于Logistic测试覆盖率函数的软件可靠性建模研究[J]. 计算机研究与发展, 2011, 48(2): 232-240.
LI Hai-feng, LI Qiu-ying, LU Ming-yan. Software reliability modeling with logistic test coverage function[J]. Journal of Computer Research and Development, 2011, 48(2): 232-240.

[6] MUSA J D. A theory of software reliability and its applications[J]. IEEE Transactions on Software Engineering, 1975, SE-1(3): 312-327.

[7] OHBA M. Software reliability analysis models[J]. IBM Journal of Research and Development, 1984, 21(4): 428-443.

[8] YAMADA S, OSAKI S. Software reliability growth modeling: Models and assumptions[J]. IEEE Transactions on Software Engineering, 1985, SE-11(12): 1431-1437.

[9] YAMADA S, OHBA M, OSAKI S. S-Shaped reliability growth modeling for software error detection[J]. IEEE Transactions on Reliability, 1983, R-32(5): 475-478.

[10] OHBA M. Software reliability analysis models[J]. IBM Journal of Research and Development, 1984, 28(4): 428-443.

[11] MORANDA P B. Predictions of software reliability during debugging[C]//The Annual Reliability and Maintainability Symposium. Washington, D. C.: [s.n.], 1975: 327-332.

[12] MORANDA P B. Event-altered rate models for general reliability analysis[J]. IEEE Transactions on Reliability, 1979, R-28(5): 376-381.

[13] MUSA J D, OKUMOTO K. A logarithmic poisson execution time model for software reliability measurement[C]//The 7th International Conference on

Software Engineering. Orlando, Florida, USA: [s.n.], 1984: 230-238.

[14] LITTLEWOOD B, VERRALL J L. A bayesian reliability growth model for computer software[J]. Applied Statistics, 1973, 22(3): 332-346.

[15] NELSON E C. A statistical basis for software reliability assessment. TRW software series report[R]. Redondo Beach, California: [s.n.], 1973.

[16] NELSON E C. Estimation software reliability from test data[J]. Microelectronics and Reliability, 1978, 17(1): 67-74.

[17] BROWN J R, LIPOW M. Testing for software reliability[C]//The International Conference on Reliable Software. Los Angeles, California, USA: [s.n.], 1975.

[18] CHEUNG R C. A user-oriented software reliability model[J]. IEEE Transactions on Software Engineering, 1980, SE-6(2): 118-125.

[19] KRISHNAMURTHY S, MATHUR A P. On the estimation of reliability of a software system using reliabilities of its components[C]//The 8th International Symposium on Software Reliability Engineering. Albuquerque, New Mexico, USA: [s.n.], 1997: 146-155.

[20] YACOUB S, CUKIC B, AMMAR H H. A scenario-based reliability analysis approach for component-based software[J]. IEEE Transactions on Reliability, 2004, 53(4): 465-480.

[21] 毛晓光, 邓勇进. 基于构件软件的可靠性通用模型[J]. 软件学报, 2004, 15(1): 27-32.
MAO Xiao-guang, DENG Yong-jin. A general model for component-based software reliability[J]. Journal of Software, 2004, 15(1): 27-32.

[22] WANG W, WU Y, CHEN M. An architecture-based software reliability model[C]//The 1999 Pacific Rim International Symposium on Dependable Computing. Hong Kong, China: [s.n.], 1999: 143-150.

[23] WANG W, PAN D, CHEN M. Architecture-based software reliability modeling[J]. Journal of Systems and Software, 2006, 79(1): 132-146.

[24] 陆文, 徐锋, 吕建. 一种开放环境下的软件可靠性评估方法[J]. 计算机学报, 2010, 33(3): 452-462.
LU Wen, XU Feng, LÜ Jian. An approach of software reliability evaluation in the open environment[J]. Chinese Journal of Computers, 2010, 33(3): 452-462.

[25] SHELDON F T, GREINER S, BENZINGER M. Specification, safety and reliability analysis using stochastic petri net models[C]//The 10th International Workshop on Software Specification and Design. Shelter Island, San Diego, California, USA: [s.n.], 2000: 123-132.

[26] YIN M, HYDE C L, JAMES L E. A petri-net approach for early-stage system-level software reliability estimation[C]//Annual Reliability and Maintainability Symposium 2000. Los Angeles, California USA: [s.n.], 2000: 100-105.

[27] ZHU L, LI Y. Reliability analysis of component software based on stochastic petri nets[C]//The 6th IEEE/ACIS International Conference on Computer and Information Science. Melbourne, Victoria Australia: IEEE, 2007: 296-301.

[28] YU R, HUANG Z. Operators for analyzing software reliability with petri net[C]//The 2008 International Symposium on Information Science and Engieering. Shanghai, China: [s.n.], 2008: 358-361.

[29] 李妍琛. 利用随机petri网分析软件可靠性[J]. 计算机应用与软件, 2009, 26(8): 133-135.
LI Yan-cheng. Analyzing software reliablity by using stochastic petri net[J]. Commputer Application and Software, 2009, 26(8): 133-135.

[30] WU Y, XIE L, LI B. Software design for reliability analysis using petri nets[C]//2010 International Conference on Measuring Technology and Mechatronics Automation. Changsha, China: [s.n.], 2010: 414-417.

[31] RAHMANI C, AZADMANESH A, SIY H. Architecture-based reliability modeling of web services using petri nets[C]//The 12th International Symposium on High Assurance Systems Engineering. San Jose, California: [s.n.], 2010: 164-165.

[32] KARUNANITHDI N, ITLEY A, MALAIY Y K. Using neural networks in reliability prediction[J]. IEEE Software, 1992, 9(4): 53-59.

[33] ADNAN W A, YAAKOB M, ANAS R, et al. Artificial neural network for software reliability assessment[C]// Tencon 2000. Kuala Lumpur, Malaysia: [s.n.], 2000: 446-451.
[34] KIRAN N R, RAVI V. Software reliability prediction using wavelet neural networks[C]//International Conference on Computational Intelligence and Multimedia Applications 2007. Sivakasi, India: [s.n.], 2007: 195-199.
[35] HARIRFOROUSH M, SEYYEDI M, MIRZAEE N. The evaluation of reliability based on the software architecture in neural networks[C]//The 3rd International Conference on Software Engineering Advances. Sliema, Malta: [s.n.], 2008: 19-24.
[36] ZHAO L, ZHANG J, YANG J, et al. Software reliability growth model based on fuzzy wavelet neural network[C]//The 2nd International Conference on Future Computer and Communication. Wuhan, China: [s.n.], 2010: 664-668.
[37] 熊小均, 梅登华. 基于改进型神经网络的软件可靠性模型[J]. 计算机工程, 2010, 36(22): 187-189.
XIONG Xiao-jun, MEI Deng-hua. Software reliability model base on improved neural network[J]. Computer Engineering, 2010, 36(22): 187-189.
[38] COSTA E O, VERGILIO S R, POZO A, et al. Modeling software reliability growth with genetic programming[C]// The 16th IEEE International Symposium on Software Reliability Engineering. Chicago, Illinois USA: IEEE, 2005: 1-10.
[39] COSTA E O, POZO A, VERGILIO S R. Using boosting techniques to improve software reliability models based on genetic programming[C]//The 18th IEEE International Conference on Tools with Artificial Intelligence. Arlington, Virginia USA: IEEE, 2006: 643-650.
[40] ZHANG Y, CHEN H. Predicting for MTBF failure data series of software reliability by genetic programming algorithm[C]//The 6th International Conference on Intelligent Systems Design and Applications. Jinan, China: [s.n.], 2006: 666-670.
[41] COSTA E O, DE SOUZA G A, POZO A T R, et al. Exploring genetic programming and boosting techniques to model software reliability[J]. IEEE Transactions on Reliability, 2007, 56(3): 422-434.
[42] AFZAL W, TORKAR R, FELDT R. Prediction of fault count data using genetic programming[C]//IEEE International Multitopic Conference 2008. Karachi, Pakistan: IEEE, 2008: 349-356.
[43] AFZAL W, TORKAR R. Suitability of genetic programming for software reliability growth modeling[C]// International Symposium on Computer Science and its Applications. Hobart, Australia: [s.n.], 2008: 114-117.

编 辑 税 红