软件可靠性模型研究进展

楼俊钢 江建慧 帅春燕 靳 昂
(同济大学计算机科学与技术系 上海 201804)

摘 要 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。对近年来提出的多种不同的软件可靠性模型进行分类剖析,讨论了部分模型的预测能力和适用性,分析了多个模型适用性差的原因,还对未来的研究趋势进行了展望。

关键词 软件可靠性模型,随机过程,未确知理论,机器学习,贝叶斯推断,混沌理论

Software Reliability Models: A Survey

LOU Jun-gang JIANG Jian-hui SHUAI Chun-yan JIN Ang
(Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract The software reliability model is one of the best approaches to predict. This parer analyzed and evaluated software reliability quantitatively. It is very important to infer software reliability by reasonably modeling and incorporating software failure data as well as other prior information. The basic concepts of software reliability models were presented. More than thirty different models proposed recently were analysed. Their predictive validity and applicability were discussed in detail. Finally, future research directions and potential applications of software reliability models were pointed out.

Keywords Software reliability models, Stochastic process, Unascertained theory, Machine learning, Bayesian inference, Chaos

可靠性是衡量所有软件系统最重要的特征之一。不可靠的软件会让用户付出更多的时间和金钱,也会使开发人员名誉扫地。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]。该概率是软件输入和系统输出的函数,也是软件中存在故障的函数,输入将确定是否会遇到所存在的故障。

软件可靠性模型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[2]。从Hudson的工作开始,到1971年Jelinski-Moranda(J-M)模型的发表,直到今天,已公开发表了几百种模型[3]。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。

软件可靠性建模可以基于规模、结构及复杂度等计量,如Halstead提出程序规模和复杂度计量[4]、McCabe提出两种程序复杂性测度[5]、分布式软件的可靠性建模[6]、基于结构的软件可靠性建模[7,8]等。本文主要关注上述第三方面的内容,所考虑的软件可靠性模型主要是基于软件失效数据,包括时间和计数数据进行建模。近年来,研究人员提出了许多基于软件失效数据的软件可靠性建模方法。按照建模理论(随机过程、混沌理论、机器学习、未确知理论及模糊理论等)的不同,可以对软件可靠性模型进行分类。

软件可靠性建模的研究尽管已经取得了一定的成效,但还存在许多关键问题没有解决,与实际项目的要求还有较远的距离。Musa[1,9]、Lyu[10]、蔡开元[5]等人已经对软件可靠性模型的研究做了比较完整的总结。在他们的基础上,本文系统地分析了软件可靠性建模方法的最新研究进展,讨论了部分模型的预测性能和适用性等,分析了多个模型通用性差的原因,最后对未来的研究趋势进行了展望。

本文在第1节对可靠性建模做了一般表述。基于随机过程模型的各种扩展方法在第2.1节中进行综合介绍。第2.2节将介绍混沌理论可靠性建模,混沌是数学科学和物理科学两栖的边缘学科,是丰富的非线性物理背景和深刻数学内涵相结合的现代学科。第2.3节将介绍软件可靠性建模领域出现的一些新的方向,如人工神经网络和支持向量机(Support Vector Machine,SVM),它们与贝叶斯推断的共同特征是研究计算机怎样模拟或实现人类的学习行为,以获取新的知识或技能,重新组织已有的知识结构使之不断改善自身的性能,这些均可归于机器学习范畴。第2.4节讨论了未确知理论用于可靠性建模。关于模糊可靠性建模,文献[5]已经有比较完整的描述,这里不再赘述。

1 软件可靠性模型的一般表述

令$(t_1,t_2,\cdots,t_n)$,$(x_1,x_2,\cdots,x_n)$分别表示失效计数数据

到稿日期:2009-10-12 返修日期:2009-12-29 本文受国家高技术研究发展计划课题(2007AA01Z142),上海市科学技术委员会信息技术领域重点科技攻关计划项目(04DZ15032和06DZ15003)资助。

楼俊钢(1982—),男,博士生,主要研究方向为软件可靠性工程、性能评估等,E-mail:loujungang0210@hotmail.com。

和失效间隔时间。基于失效数据的可靠性模型的主要作用有两个：一个是失效过程的描述，通过分析已知失效数据，总结出统计规律，找到数据内在的相互依赖关系；另一个是对 t_{n+d}，x_{n+d}（分别表示第 $n+d-1$ 次和 $n+d$ 次失效之间的时间间隔、第 $n+d$ 个时间段的失效次数）做出合理的预测。但软件失效具有不确定性，这是因为软件系统的开发过程、软件自身的复杂性、软件操作剖面、软件失效行为及调试过程多样化使得人们很难对其进行精确刻画，因此研究人员采用随机性、混沌性、模糊性、未确知理论等来描述软件失效的不确定性。

随机过程模型物理意义明确，一般包括 3 个过程，即学习特殊→普遍理论→判断未知，因此可以建立较为完善的可靠性指标体系。能够计算得到的可靠性指标包括均值函数、失效率函数、下一阶段的失效期望数、可靠度、累积分布函数、失效密度函数及 MTTF 等。而其它模型则不同，它们不需要先从已知的数据中总结出普遍真理，然后再推出未知，而是通过之前的学习，直接可以判断未知，因此一般只具有预测功能。能够计算的可靠性指标通常只包括未来阶段失效次数或软件未来失效时间等。

2 软件可靠性模型分类

2.1 基于随机过程的模型

随机过程可靠性模型是软件可靠性增长模型中最重要的一类，也是当前软件可靠性增长模型领域研究最多、实际项目中应用最广泛的一类，在很多应用中都有良好的表现。随机过程可以描述随时间变化的软件失效行为，并具有这样的性质：过程的将来行为仅依赖于现在的状态，而与它的历史无关。随机过程模型的主要优点在于其模型的简单性、易于理解和实施。按照到时间 t 的累计失效次数服从的分布，随机过程可靠性模型主要可以分为泊松过程模型和二项式模型。模型的一般形式可以表示为

泊松：

$$P\{m(t_{i+1})-m(t_i)=n\}=\frac{e^{-[m(t_{i+1})-m(t_i)]}[m(t_{i+1})-m(t_i)]^n}{n!},$$

$$m(t)=\int_0^t \lambda(x)\mathrm{d}x$$

二项式：

$$P_m(t)=\binom{\mu_0}{m}[F_a(t)]^m[1-F_a(t)]^{\mu_0-m},\lambda(t)=\mu_0 F_a'(t)$$

随机过程模型的参数一般有比较明确的物理意义。如在均值函数为 $m(t)=a(1-e^{-bt})$ 的 G-O 模型中，参数 a 表示测试开始软件中所包含的初始故障总数，而参数 b 表示故障检测率（Fault Detection Rate，FDR）。G-O 模型假设故障总数和故障检测率为常数[1]。而一般情况下的故障总数及故障检测率可以表示为关于时间 t 的函数 $a(t)$，$b(t)$。正因为这些明确的物理意义，在建立随机过程模型时，对研究的对象做一个精确、无歧义的假设是有必要的。实际的软件失效过程非常复杂，必须进行抽象和简化、确定边界条件，然后才可用数学理论和工具来对它进行研究。假设的不同会导致模型的千差万别，假设是可靠性建模的前提和基础，而符合实际的假设是提高可靠性模型精度的保证。近年来，随机过程软件可靠性模型的大部分研究工作是模型的统一、对已有模型的改进、可靠性成本模型等。模型改进的出发点是提出更加合理的假设，以提高模型预测精度，如考虑测试环境与实际运行环境的差别、故障的相关性、不完美调试、测试者学习能力、测试效用函数（Testing Effort Function，TEF）等。

2.1.1 模型的统一

Huang 和 Lyu 等人通过使用加权算术、几何、调和平均数等，提出了非齐次泊松过程模型（NHPP）的统一框架[11,12]。目前存在的所有 NHPP 模型，如 G-O 模型、对数曲线增长模型、几何曲线增长模型、Duane 模型、延迟 S-曲线增长模型等均可以用统一框架经过适当的变换得到，并且可以推导出一些新的 NHPP 模型。它们的方法把许多 NHPP 模型统一为

离散的情形：

$$m(i)=g^{-1}(\omega^i g(m(0))+(1+\omega^i)g(a))$$

$$m(i)=g^{-1}(\mu_i g(m(0))+(1-\mu_i)g(a))$$

连续的情形：

$$m(t)=g^{-1}(g(a)+\{g(m(0))-g(a)\}\exp[-\int_0^t b(\mu)\mathrm{d}\mu] \tag{1}$$

2.1.2 测试环境与运行环境的差别

早期的绝大多数软件可靠性模型假设软件的测试环境和操作环境是相似的，利用测试获得的软件失效信息对软件系统的失效过程进行建模，评估软件系统的可靠性，预测软件实际工作时的现场行为。然而，由于实际运行环境的复杂性，软件的测试剖面很难真实地反映运行剖面。在可靠性模型中，表现测试环境与运行环境不同的关键是找到合适的方法来描述故障检测率的不同。

Musa 最早提出用压缩因子来表示测试时失效率与运行时失效率的比率[2]；Yang 和 Xie 假设运行环境下失效率接近于测试时的软件失效率，但运行时失效率随着时间呈现下降的趋势，而测试时失效率为常数[13]。

Zhang，Jeske 和 Pham 定义了压缩因子 K_c，K_d，使其表述更加精确[14-17]。用 $b(t)$ 表示在时间 t 的故障检测率，$b_{test}(t)$，$b_{feild}(t)$ 分别代表测试和现场环境下的故障检测率。$\bar{b}_{test}=\lim_{T\to\infty}\frac{1}{T}\int_0^T b_{test}(t)\mathrm{d}t$，$\bar{b}_{test}$ 定义了整个测试阶段的平均故障检测率。同样可以定义 $\bar{b}_{feild}$，这样就有 $K_d=\bar{b}_{test}/\bar{b}_{feild}$。$K_c$ 表示测试结束时故障总数 $\bar{a}$ 与实际运行开始时故障总数 C 的比率，即 $K_c=\bar{a}/c$。针对两组失效数据，他们采用假设检验的方法证明 $K_c=1$，$K_d=1$ 均是错误的。适当的 K_c，K_d 值可以由软件早期版本或类似软件得到。

赵靖、刘宏伟等人认为软件测试阶段的 FDR 无论是哪种情况，环境因子都应是随着时间变化的函数[18]。为了近似描述变化的环境因子，定义 $k(t)=b_{test}(t)/b_{feild}(t)$。为了近似地描述环境因子的变化趋势，根据实测数据，用离散的随时间变化的均值来描述环境因子的变化趋势。随时间变化的环境因子均值定义为 $\bar{K}(t)=\bar{b}_{test}(t)/\bar{b}_{feild}(t)$。

Teng 和 Pham 等人采用环境因子随机变量 η 来反映软件运行阶段各种随机因素对软件失效率的影响[19]，定义 $\eta=\begin{cases}1, t\leqslant T\\ r.v.\ with\ \cdot\ p.d.f.\ f(\eta), t\geqslant T\end{cases}$，其中，$T$ 表示测试结束时间。他们以 η 服从 β，γ 分布情况为例，说明了随机现场模型（Random Field Environment，RFE）的具体形式。

Huang，Kuo，Lyu 等人用映射 $T:R\to R$ 来表示从测试到

运行阶段故障检测率的变换[20,21]。在测试阶段，故障检测率为常数的情况下，推荐了4种故障检测率变换函数：$T(s|b)=b$，$T(s|b)=bcs^{c-1}$，$T(s|b)=\frac{b}{1+c\exp(-bs)}$，$T(s|b)=\frac{b^2 s}{1+bs}$。其中，$s$ 表示时间间隔。结合式(1)，可以计算得到运行阶段的均值函数。

2.1.3 不完美调试及测试者学习过程

早期可靠性模型往往假设一旦发生失效，引起失效的故障就会立刻被检测并消除，且不会引入新的故障。而实际上在修复故障的过程中还可能会引入新的故障。在测试阶段，有故障的修复百分比在商业软件开发组织中从1%或2%到大于10%[2,8]。Geol 和 Okumoto 最早考虑到不完美调试的可靠性建模问题，Ohba 和 Yamada 等人也做了很多工作[8]。另外，随着软件测试过程的进行，测试人员对软件越来越了解，发现故障的能力也越来越强，这反映了测试者的学习过程。许多学者建议可靠性建模时应该考虑这一因素。

Zhang，Teng 和 Pham 等人假设软件调试时以故障引入率(Fault introduction Rate) $\beta(t)\ll p$ 的概率引入新的故障[22-24]，P 表示故障排除效率(Fault Removal Efficiency)，即故障排除百分比。模型的均值函数为

$$m(t)=a\int_0^t b(u)\exp\left(-\int_0^u (p-\beta(\tau))b(\tau)\mathrm{d}\tau\right)\mathrm{d}u$$

Pham，Nordmann 和 Zhang 等人假设软件调试时会以概率 α 引入新的故障[25,26]。α 为常数，结合测试者的学习过程，则软件故障总数及故障检测率函数分别为

$$a(t)=a(0)(1+\alpha\cdot t),$$

$$b(t)=\frac{b(0)(1+\beta)}{1+\beta\cdot\exp[-b(0)(1+\beta)\cdot t]}$$

式中，$b(0)$ 表示故障初始值。

刘宏伟和杨孝宗等人提出了结合测试者学习过程和软件固有 FDR 的可靠性模型[27]。他们采用了文献[22]中的学习过程数学模型，认为软件中隐藏的故障被检测到的概率并不相同，容易检测到的故障首先被检测出来并排除掉。随着测试的进行，剩余的故障平均被检测到的概率会降低。如果没有其它因素的影响，FDR 应是一个递减的函数。并且经过了足够长的时间还没被发现的故障，其被检测到的概率趋近于0。软件 FDR 的这种变化趋势可表示为 $P_0(t)=b(0)\cdot\exp(-\beta_1\cdot t)$，从而得到 $b(t)=\frac{b(0)(1+\beta)\exp(-\beta_1\cdot t)}{1+\beta\cdot\exp[-b(0)(1+\beta)\cdot t]}$。在此工作基础上，赵靖、刘宏伟等人考虑了测试与运行阶段的差别[28]。在他们所建立的模型中，测试阶段故障检测率与文献[27]相同，而运行阶段故障检测率为 $b(t)=b^0\exp(-\beta_2\cdot t)$。

2.1.4 故障相关性

在绝大部分可靠性模型中，很重要的一个假设是各故障是独立分布的。但一些新的研究表明，在软件的测试过程中，故障之间的相关性是普遍存在的。例如，测试者所使用的一系列测试手段并不总是随机的，测试的非随机性可以用动态的故障相关性表示。Wu Kang 等人把故障分为3类，并推导出各种情况下的故障检测率函数[29]。Katerina 等人建立了考虑故障相关性的、基于马尔可夫更新过程的软件可靠性增长模型[30]，模型考虑了相关性对软件可靠性的影响，分为成功和失效两种状态。在此基础上，Dai Yuan Shun 等人结合失效严重等级，把模型扩展为有 n 种状态的情形[31]。Huang Chin Yu 等人在建立模型时，将软件故障之间的相关性分为两类，一类是独立的故障，另一类是相关的故障。然后对这两类故障分别建模[32,33]。赵靖等人使用有向图对故障相关性做了形式化描述，把故障之间的相关关系分为3类：一对一、一对多、多对一，并结合测试与运行阶段的差别，建立了可靠性模型[34]。

2.1.5 测试效用函数

一般情况下，测试资源的消耗量与时间不成比例。测试效用函数描述测试资源随着时间变化的消耗分布情况[35,36]，基于 TEF 的软件可靠性模型可以描述测试资源消耗各种分布情况下的测试效率，具体来讲就是测试资源消耗不同的分布对故障暴露时间的影响，其均值函数为[37,38]。

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

在给定边界条件 $m(0)=0$，并且 $b(t)=b$ 的情况下，可以得到 $m(t)=a(1-\exp[-b[W(t)-W(0)])$。其中，$w(t)$ 表示时间 t 的测试资源消耗量；$W(t)$ 表示到时间 t 的累积测试资源消耗量。Huang 等人得出 $m(0)=0$，并且 $b(t)=\frac{b^2 t}{1+bt}$ 的情况下，$m(t)=a\{1-[1+b(W(t)-W(0))]\exp[-b(W(t)-W(0))]\}$[39,40]。表1总结了现有的 TEF。

表1 测试效用函数

作者	TEF	w(t)或 W(t)	备注
Chatterjee 等人[41]	Chatterjee TEF	$w(t)=\frac{k}{[f(t)]^p}$	K 表示比例常数，f(t)是学习因子
Pillai，Nair[42]	Pillai and Nair Gamma Model	$w(t)=\frac{4N}{t_d^3}\cdot t^2\cdot\exp(-\frac{2t}{t_d})$	N 表示总消耗的测试资源
Tohma 等人[43,44]	Tohma Test-Instance Functions	$w(t)=tester(t)\cdot(a\cdot t+b)$ $w(t)=tester(t)\cdot(a\cdot t^2+b\cdot t+c)$ $w(t)=[tester(t)]^p\cdot a$ $w(t)=p_{LT}\cdot[1-\exp(-\mu_i\cdot a\cdot i)]$，$a>0,0<p_{LT}\leqslant 1$	tester(t)表示时间 t 时的测试人员总数
Hou，Kuo[45]	Hou and Kuo Learning-Factor Functions	$w(t)=\frac{p_{LT}}{1+b\cdot\exp(-\mu_i\cdot a\cdot i)}$， $a>0,b>0,0<p_{LT}\leqslant 1$	p_{LT} 表示学习因子极限值
Yamada[35,46]	Weibull-Type TEF	$W(t)=N\cdot[1-\exp(-\beta\cdot t)]$ $W(t)=N\cdot[1-\exp(-\frac{\beta}{2}\cdot t^2)]$ $W(t)=N\cdot[1-\exp(-\beta\cdot t^m)]$	N 表示总消耗的测试资源
Huang 等人[39,40]	Logistic TEF	$W(t)=\frac{N}{1+A\cdot\exp(-\alpha\cdot t)}$	N 表示总消耗的测试资源

2.1.6 软件可靠性成本模型

何时停止测试、如何分配测试资源以及估计软件的可靠性水平，是软件开发和测试人员最关心的问题之一。一系列可靠性成本模型的提出，为测试成本的优化提供了较好的参

考[47-62]。

可靠性成本模型主要考虑测试成本、故障移除成本、软件失效带来的损失等与经过可靠性测试节省的调试成本之间的平衡，以此来确定软件最优发布时间。软件最优发布时间 T^* 可以看作在适当的约束条件下非线性规划问题的最优解。

$$E(T)=C_0+E_1(T)+E_2(T)+E_3(T)+E_4(T)$$

式中，C_0 为一固定值数，表示软件测试启动费用；$E_1(T)$，$E_2(T)$，$E_3(T)$，$E_4(T)$分别表示总的测试费用、测试时故障移除总费用、运行时故障移除总费用、软件失效引起的风险费用，在不同的模型中有不同的函数形式。

2.1.7 其它随机过程模型

Ishii 和 Tadashi 提出了把日历时间和执行时间作为双变量的二维可靠性模型[63]。邹丰忠和李传湘等人提出了双随机软件可靠性模型[64]，他们认为若要全面地去表述软件的失效行为，就需要一个补偿随机过程，并且不对软件失效过程的概率分布、强度曲线做任何假设。Dai，Xie Min 等人采用了马尔可夫更新过程[31]；Pham，Wang 等人采用了拟更新过程[65,66]；W. C. Huang 和 C. Y. Huang 等人采用了排队论[67]；Simon 等人采用了顺序统计量[68]；Wen 和 Tohmas 采用了移动平均数对失效数据建模[69]；Lyu 建立了 N 版本软件可靠性模型[70]。

2.1.8 小结

目前，很多软件可靠性模型方面的研究都是对 NHPP 模型的改进，这主要是因为 NHPP 模型在描述软件失效时有独一无二的物理意义上的优势。看法不一致的地方主要在模型的均值函数的描述上。使用不同假设时，可以得到不同的均值函数，但目前还没有证据能表明不同的软件失效过程是否可以或者不可以用同样的均值函数来描述。

随机过程模型揭示了软件失效的统计规律，但它只陈述软件失效数据在各个场合中取值的概率分布，只给出数目众多的一类实例中失效次数的平均值，并不预言任意一个软件会出现怎样的失效。对软件失效过程来讲，这一过程是无法重复的，是单个事件，因此预测时会存在很大的偶然性。

在不同数据集上使用随机过程模型时，预测精度往往差别较大。到目前为止，还没有理论或实验能说明这种差别的来源。可能的原因，一是不同的软件有不同的失效均值过程；二是同一随机过程两个不同的样本有可能符合不同的曲线，而相关研究或者实际工程应用中所应用的软件可靠性失效数据实际上仅仅是软件失效过程的一个样本轨迹。

随机过程可靠性模型研究可能的发展方向包括：

第一，模型求解方法的研究。软件失效的复杂性和输入的不确定性造成软件模型复杂度越来越高，难于求解。越来越多的复杂模型的建立使得如何求解大规模的随机模型和建立可求解的可靠性模型成为未来的研究方向之一。

第二，对软件可靠性的预测需要考虑多种错综复杂的因素，有些是基本因素，有些是偶然因素。在不同的假设条件下，采用不同的单种预测方法对同一预测问题建立多种预测模型，从众多的模型中选择“最好”的一个未必是提高预测精度最佳的办法。不同的预测模型方法各有优点和缺点，它们之间并不是相互排斥，而是相互联系，相互补充。若预测者只用一种预测方法进行预测，则这种方法的选择是否适当就显得很重要。在不能很好地判断何为最优预测方法的情况下，把多种单项预测方法正确地结合起来使用，会使得组合预测结果对某个较差的预测方法不太敏感，因此它一般能提高预测精度。

第三，把软件失效过程当成随机过程的一个具体实现，而不是均值过程来看待。均值过程反映了很多实验的统计行为，而同一软件的失效过程具有不可重复性，因此把软件失效过程作为一个具体实现来建模，更加符合软件失效行为的特征。

第四，在软件测试及运行中，收集失效数据周期长，耗费大，因此通过故障注入[71]让软件加速失效，以此来寻找失效与可靠性之间的关系，建立基于故障注入的可靠性模型，值得进行研究。

2.2 混沌理论

邹丰忠和李传湘等人采用混沌时间序列预测中常用的基于实际观测数据的相空间重构方法来预测软件的下次失效时间[72,73]。

混沌预测方法就是在相空间中找到一个非线性模型去逼近失效数据序列的动态特性。采用混沌理论，首先需要判断软件失效数据是否具有混沌性。判别方法是计算最大 Lyapunov 指数是否大于零。在判断失效序列具有混沌性后，不失一般性，设$\{X(t),t=0,1,2,\cdots,n\}$表示要研究的混沌失效时间序列，选择适当的时间延迟 τ 和嵌入维数 m 对其重构的相空间为 $X(t)=\{x(t),x(t-\tau)\cdots,x[t-(m-1)\tau]\}$。由 Takens 定理知，只要 τ 和 m 选择恰当，则存在一个光滑映射 $F:R^m\to R$，使得 $X(t+\eta)=F\{x(t),x(t-\tau),\cdots,x[t-(m-1)\tau]\}$，$\eta$ 为预测步长。理论上 F 是唯一的，但是实际中可用数据总是有限的，求得真正的 F 很困难，只能充分逼近 F。

基于混沌理论的可靠性模型的研究还处于起步阶段。在软件可靠性预测中采用相空间重构预测及其它混沌预测方法还有许多问题需要解决，如需要的数据量大、参数选择难度大、抗噪性能差、预测性能不稳定、高阶情况待定参数多等问题。

2.3 未确知理论

用未确知理论进行可靠性建模由张勇强和孙胜娟提出[74]，其主要思想是对可以获得的部分系统输出数据（失效数据）进行分析，将其蕴涵的系统失效特征用未确知的有理数表达出来，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。该模型不对软件故障过程做任何统计规律的分布假设，更真实地描述了软件失效的特征，理论上具有较高的预测精度和较好的适用性。在他们的模型中，从第 $i-1$ 次失效到第 i 次失效发生的时间 x_i 的可靠性函数为

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

2.4 机器学习

根据给定的训练样本，机器学习的目的是求出对某系统输入输出之间依赖关系的估计，使它能够对未知输出做出尽可能准确的预测。

基于机器学习的可靠性模型一般可表示为变量 y 与 x 存在一定的未知依赖关系，根据 l 个独立同分布观测样本$(x_1,y_1),(x_2,y_2),\cdots,(x_l,y_l)$，在一组函数$\{f(x,w)\}$中求一个最优的函数 $f(x,w_0)$ 对依赖关系进行估计，使期望风险最小。

其中，x 表示已知的软件失效数据，y 表示将来的失效数据，$\{f(x,w)\}$表示预测函数集，w 为广义参数。

贝叶斯模型的基本思想是假设可靠性数据的分布 $f_T(t|\xi)$依赖于未知参数 ξ，$g(t;\xi)$是反映 ξ 历史数据的先验分布[75-78]。ξ随着收集到数据集的变化而变化，并反映在ξ的后验分布中。ξ的估计可以由它的后验分布得到，从而得到与 ξ 相关的可靠性参数。采用贝叶斯推断进行可靠性建模，可以充分利用样本信息和参数的先验信息，在进行参数估计时，具有更小的方差或平方误差，得到更加精确的预测效果。采用贝叶斯推断进行可靠性建模主要应考虑以下两个方面的问题：

(1)先验分布的选择

先验分布是贝叶斯推断的基础和出发点。参数先验分布的选取方法包括贝叶斯假设、基于信息函数的选择方法、概率匹配先验分布、最大熵原则、相对似然函数法、积累函数法、蒙特卡罗法、Harr 不变测度法、随机加权法、Bootstrap 法等。

(2)后验分布推断

在贝叶斯分析中，一般只能得出后验分布密度函数的核，难以获得具体的密度函数。在 20 世纪 90 年代前，解决这个问题的方法主要有 Lindly 数值逼近法、Naylor-Smith 逼近法、Tierney-Kadane 逼近法等。然而，这些方法的实现需要依靠复杂数值和解析方法。马尔可夫链蒙特卡罗法(MCMC)的出现为贝叶斯推断理论和方法的应用开辟了广阔的情景，它主要包括 Gibbs 抽样方法和 Metroplis-Hastings 方法。

Karunanith 等人最早把人工神经网络理论应用到软件可靠性评估和预测中[79-81]。把累积执行时间序列集合 $I_k(t)$及其相应的累计失效次数集合 $O_k(t)$作为输入，使用神经网络映射 $P:((I_k(t),O_k(t)),i_{k+b}(t+\Delta t))\mapsto O_{k+b}(t+\Delta t)$对第 $k+d$ 时间段内的累计失效次数进行预测。所使用的网络包括采用 C-C 学习算法的前向神经网络模型、采用 BP 算法的 Elman 循环网络模型、采用 C-C 学习算法的 Jordan 半级联神经网络模型。Su Yu-Shen 等人则使用神经网络给不同的 NHPP 模型赋予不同的权值，从而构造了一个综合模型[82]。Aljahdali[83]、蔡开元[84,85]、Ho[86]、Sitte[87]、Tian 和 Noore-1[88,89]等人也各自做了一些工作。

通过学习，神经网络可以根据不完全、复杂、非确定性的数据集获得输入和输出之间的非线性映射，配合适当的实现技术，使用神经网络预测可靠性，可以取得较好的效果。但是，其计算复杂性随神经元个数的增长而快速增长。因此，寻找合适的网络，采用合理的网络规模，提高检测准确性，降低计算复杂性等，是主要的研究方向。神经网络可靠性模型的另外一个问题是容易出现过度拟合。

SVM 是 Vapnik 提出的一类新型机器学习方法[90]。它建立在统计学习的维理论(Vapnik-Chervonenkis，VC)和结构风险最小化(Structural Risk Minimization，SRM)原理基础上。SVM 的关键在于核函数，低维空间向量集通常难于划分。解决的方法是将它们映射到高维空间，但这个办法带来的困难就是增加了计算复杂度，而核函数正好巧妙地解决了这个问题。也就是说，只要选用适当的核函数，就可以得到高维空间的分类函数。

Tian 和 Noore 等人率先把 SVM 引入到软件可靠性建模当中[91]。他们使用 SVM 来描述软件失效时间之间的内在关系。SVM 的学习随着每次失效的发生动态调整，并捕捉隐藏在软件失效时间序列中最重要的性质。Yang 等人提出的模型一般步骤为：确定 w 值→选择核函数进行学习→预测[92]。

SVM 最大的优势是根据有限的样本信息在模型的复杂性和学习能力之间寻求最佳折衷，尽量提高学习机的泛化能力，较好地解决小样本、高维数、非线性和局部最小点等问题。

机器学习软件可靠性模型虽然在预测精度上比随机过程模型有较大的优势，但存在的主要问题是模型仅具有预测功能，并不能对软件失效过程做出一个很好的解释。因此，需要研究如下两类方法：

(1)机器学习与随机过程相结合。两类方法具有互补性，把它们有机地结合起来，发挥各自的优势。

(2)基于机器学习的混沌时间序列预测方法。近些年来，基于前馈神经网络、自组织特征映射、核方法和 SVM、有限脉冲神经网络、储备池的混沌时间序列预测方法已经在生物学、生理学、数学、物理、天文气象、经济学等领域得到了成功的应用。可靠性预测与其中的很多预测问题是类似的，把这些方法应用于可靠性中可能会得到很好的效果。

结束语 动态规划法是研究优化问题的一种常用算法，满足最优性原理是使用动态规划法的前提条件，因此，许多文献用最优性原理的证明过程分析最优解的求解过程。本质上说，文献[19]是这样思考 Hamilton 回路的。文献[21]也是这样讨论 TSP 的。本文的研究结果表明，最优性原理本身并没有提供最优的解结构，具体而言：对于 TSP 问题，虽然一条长度为 n 的最短路径一定是一长度为 1 的路径和一长度为 $n-1$ 的路径的最短路径组合，然而，这种组合未必是求解长度为 n 的路径的最优过程，结合肖尔茨猜想的证明，本文探讨了最优解组成的复杂性，为进一步的研究提供了理论基础。从工程应用而言，在 PC 机环境下寻找中小规模的 TSP 最优算法具有广泛的应用价值，文献[22]研究指出：普通枚举算法适应规模为 10 个点以下，完全贪心算法适应规模为 15 个点以下，文献[23]提出的验证算法适应规模为 20 个点以下，比较而言，本文的算法适应规模为 32 点以下，特点如下：

1. 计算不依赖于初始解与经验值。

2. 没有欧氏空间的平面约束条件，有利于推广到其他相关非平面的应用领域。

3. 作为一种精确算法，其解可以作为近似算法的评价依据。

4. 在降低时间复杂度的同时，为以后优化留下了很大的研究空间——数据压缩与矩阵并行运算的相关研究将是该算法有潜力的发展方向。

参 考 文 献

[1] IEEE Recommended Practice on Software Reliability[S]. IEEE Std 1633-2008

[2] Musa J D. Software reliability engineering [M]. New York: McGraw Hill, 1999

[3] Whittaker J A, Voas J. Toward a more reliable theory of software reliability[J]. IEEE Computer, 2000, 13(12): 36-42

[4] Halstead M H. Elements of software science [M]. Elsevier: North Holland, 1975

[5] 蔡开元. 软件可靠性工程基础[M]. 北京：清华大学出版社，1995

[6] Chiu C C, Hsu C H, Yeh Y S. A Genetic Algorithm for Reliabili-

ty-oriented Task Assignment with k Duplications in Distributed Systems[J]. IEEE Transactions on Reliability, 2006, 55(1): 105-117

[7] Swapna S G, Kishor S T. Analytical Models for Architecture-based Software Reliability Prediction: A Unification Framework [J]. IEEE Transactions on Reliability, 2006, 55(4): 578-590

[8] Swapna S G. Architecture-based Software Reliability Analysis: Overview and Limitations[J]. IEEE Transactions on Dependable and Secure Computing, 2007, 4(1): 32-40

[9] Musa J D, Okumoto K. Software reliability models: concepts, classification, comparisons, and practice [M]. Heidelberg: Springer Verlag, 1983

[10] Lyu M. Handbook on software reliability engineering[M]. New York: McGraw Hill, 1996

[11] Huang C Y, Lyu M R, Kuo S Y. A Unified Scheme of Some Nonhomogenous Poisson Process Models for Software Reliability Estimation[J]. IEEE Transactions on Software Engineering, 2003, 29(3): 261-269

[12] Inoue S, Yamada S. Generalized Discrete Software Reliability Modeling With Effect of Program Size[J]. IEEE Transactions on Systems, Man and Cybernetics, Part A, 2007, 37(2): 170-179

[13] Yang B, Xie M. A study of operational and testing reliability in software reliability analysis[J]. Reliability Engineering and System Safety, 2000, 70: 323-329

[14] Zhang X M, Teng X L, Pham H. Considering fault removal efficiency in software reliability assessment[J]. IEEE Transactions on Systems, Man and Cybernetics—Part A, 2003, 33(1): 114-120

[15] Zhang X M, Jeske D R, Pham H. Calibrating software reliability models when test data does not reflect the user operational profile[J]. Applied Stochastic Models in Business and Industry, 2002, 18: 87-99

[16] Teng X L, Pham H. A software cost model for quantifying the gain with considerations of random field environments[J]. IEEE Transactions on Computers, 2004, 53(3): 380-384

[17] Jeske D R, Zhang X M, Pham D. Adjusting software failure rates that are estimated from test data[J]. IEEE Transactions on Reliability, 2005, 54(1): 107-114

[18] 赵靖, 刘宏伟, 崔刚, 等. 考虑测试环境与实际运行环境的软件可靠性增长模型[J]. 计算机研究与发展, 2006, 43(5): 881-887

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

[20] Huang C Y, Kuo S, Lyu M R, et al. Quantitative software reliability modeling from testing to operation[C]// Proc. 11th Int. Symp. Software Reliability Engineering, 2000: 72-82

[21] Huang C Y, Lyu M R. Optimal testing resource allocation and sensitivity analysis in software development[J]. IEEE Transactions on Reliability, 2005, 54(4): 592-603

[22] Zhang X, Teng X L, Pham H. Considering fault removal efficiency in software reliability assessment [J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2003, 33(1): 114-120

[23] Pham L, Pham H. Software reliability models with time-dependent hazard function based on Bayesian approach [J]. IEEE Trans. Syst., Man, Cybernetics-Part A: Systems and Humans, 2000, 30(1): 25-35

[24] Pham H. Software Reliability[M]. New York: Springer-Verlag, 2000

[25] Pham H, Nordmann L, Zhang X M. A General Imperfect-Software-Debugging Model with S-shaped Fault-detection Rate[J]. IEEE Transactions on reliability, 1999, 48(2): 169-176

[26] Zhang X M, Pham H. An analysis of factors affecting software reliability[J]. Journal of Systems and Software, 2000, 50(1): 43-56

[27] 刘宏伟, 杨孝宗, 曲峰, 等. 一个基于响铃形故障检测率函数的软件可靠性增长模型[J]. 计算机学报, 2005, 28(5): 908-913

[28] 赵靖, 刘宏伟, 崔刚, 等. 考虑测试与运行差别的软件可靠性增长模型[J]. 计算机研究与发展, 2006, 43(3): 503-508

[29] Wu K, Malaiya Y K. The Effect of Correlated Faults on Software Reliability[C]// Proceedings of the 4th International Symposium on Software Reliability Engineering. Denver, 1993: 80-89

[30] Goseva P K, Trivedi K S. Failure correlation in software reliability model[J]. IEEE Trans. Reliability, 2000, 49(1): 37-48

[31] Dai Y S, Xie M, Poh K L. Modeling and Analysis of Correlated Software Failures of Multiple Types[J]. IEEE Trans. Reliability, 2005, 54(1): 100-107

[32] Huang C Y, Chu T L. Software Reliability Analysis by Considering Fault Dependency and Debugging Time Lag [J]. IEEE Trans. Reliability, 2006, 55(3): 436-451

[33] Lo J H, Huang C Y. An integration of software failure detection and fault correction processes in software reliability analysis[J]. Journal of Systems and Software, 2006, 32(1): 32-41

[34] 赵靖, 张汝波, 顾国昌. 考虑故障相关的软件可靠性增长模型研究[J]. 计算机学报, 2007, 30(10): 1713-1720

[35] Yamada S, Hishitani J, Osaki S. Software reliability growth model with Weibull testing effort: a model and application[J]. IEEE Trans. Reliability, 1993, 42(1): 100-105

[36] Wu Y P, Hu Q P, Xie M. Modeling and Analysis of Software Fault Detection and Correction Process by Considering Time Dependency[J]. IEEE Transactions on Software reliability, 2007, 56(4): 629-642

[37] Huang C Y, Kuo S Y. Analysis and assessment of incorporating logistic testing effort function into software reliability modeling [J]. IEEE Trans. Reliability, 2002, 51(3): 261-270

[38] Kuo S Y, Huang C Y, Lyu M R. Framework for modeling software reliability, using various testing-efforts and fault-detection rates[J]. IEEE Trans. Reliability, 2001, 50(3): 310-320

[39] Huang C Y, Kuo S Y. Analysis of incorporating logistic testing effort function into software reliability modeling [J]. IEEE Trans. on Reliability, 2002, 51(3): 261-270

[40] Huang C Y, Kuo S-Y, Lyu M R. An Assessment of Testing-Effort Dependent Software Reliability Growth Models[J]. IEEE Transactions on Software reliability, 2007, 56(2): 198-211

[41] Chatterjee R, Misra B, Alam S S. Joint effect of test effort and learning factor on software reliability and optimal release policy [J]. Inter'l. J. Systems Science, 1997, 28(4): 391-396

[42] Pillai K, Nair V S S. A model for software development effort and cost estimation [J]. IEEE Trans. Software Engineering, 1997, 23(8): 534-542

[43] Tohma Y, Tokunaga K, Nagase S, et al. Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution[J]. IEEE Trans. Software Engineering, 1989, 15(3): 345-355

[44] Tohma Y, Jacoby R, Murata Y, et al. Hyper-geometric distribution model to estimate the number of residual software faults

[C]// Proc. COMPSAC. 1989:610-617

[45] Hou R H, Kuo S Y, Chang Y P. Applying various learning curves to hyper-geometric distribution software reliability growth model[C]// Proc. 5th Int. Symp. Software Reliability Engineering. 1994:8-17

[46] Yamada S, Hishitani J, Osaki S. Software reliability growth modeling: Models and applications[J]. IEEE Trans. Software Engineering, 1985, 11(12):1431-1437

[47] Ohtera H, Yamada S. Optimal allocation and control problems for software-testing resources[J]. IEEE Transactions on Reliability, 1990, 39(2):171-176

[48] Dai Y S, Xie M, Poh K L, et al. Optimal testing resource allocation with genetic algorithm for modular software systems[J]. Journal of Systems and Software, 2003, 66(1):47-55

[49] Helander M E, Zhao M, Ohlsson N. Planning models for software reliability and cost[J]. IEEE Transactions on Software Engineering, 1998, 24(6):420-434

[50] Huo R H, Kuo S Y, Chang Y P. Optimal release times for software systems with scheduled delivery time based on HGDM[J]. IEEE Trans. on Computers, 1997, 46(2):216-221

[51] Hou R H, Kuo S Y, Chang Y P. Needed resources for software module test, using the hyper-geometric software reliability growth model[J]. IEEE Transactions on Reliability, 1996, 45(4):541-549

[52] Lyu M R, Rangarajan S, Moorsel A P A. Optimal allocation of test resources for software reliability growth modeling in software development[J]. IEEE Transactions on Reliability, 2002, 51(2):183-192

[53] Yang B, Xie M. Optimal testing-time allocation for modular systems[J]. International Journal of Quality and Reliability Management, 2001, 18(8):854-863

[54] 刘宏伟，杨孝宗，曲峰，等. 基于 CGOM 的软件费用模型研究[J]. 计算机学报，2003，26(10)：1333-1336

[55] Zhang X M, Pham H. A software cost model with error removal times and risk costs[J]. International Journal of Systems Science, 1998, 29(4):435-442

[56] Zheng S H. Dynamic release policies for software systems with a reliability constraint[J]. IIE Transactions, 2002, 34(3):253-262

[57] Huang C Y, Lyu M R. Optimal Release Time for Software Systems Considering Cost[J]. Testing-Effort and Test Efficiency, 2005, 54(4):583-591

[58] Huang C Y, Lyu M R. Optimal Testing Resource Allocation, and Sensitivity Analysis in Software Development[J]. IEEE Transactions on Reliability, 2005, 54(4):592-603

[59] Pham H, Zhang X M. A Software Cost Model with Warranty and Risk Costs[J]. IEEE Trans. Computers, 1999, 48(1):71-75

[60] Teng X L, Pham H. A Software Cost Model for Quantifying the Gain with Considerations of Random Field Environments[J]. IEEE Transactions on Computers, 2004, 53(3):380-384

[61] Zhang X M, Teng X L, Pham H. Considering Fault Removal Efficiency in Software Reliability Assessment[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2003, 33(1):114-120

[62] Yang B, Hu H J, Jia L X. A Study of Uncertainty in Software Cost and Its Impact on Optimal Software Release Time[J]. IEEE Transactions on Software Engineering, 2004, 34(6):813-825

[63] Tomotaka I, Tadashi D. Two-dimensional Software Reliability Models and Their Applications[C]// Proc. 12th Pacific Rim International Symposium on Dependable Computing. 2006:3-10

[64] 邹丰忠，李传湘. 双随机软件可靠性模型的建模[J]. 软件学报，1998，9(1)：69-73

[65] Pham H, Wang H Z. A Quasi-renewal Process for Software Reliability and Testing Costs[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2001, 31(6):623-631

[66] Pham S, Pham H. Quasi-renewal Time-delay Fault-removal Consideration in Software Reliability Modeling[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2009, 39(1):1-10

[67] Huang C Y, Huang W C. Software Reliability Analysis and Measurement Using Finite and Infinite Server Queueing Models [J]. IEEE Transactions on Software reliability, 2008, 57(1):192-203

[68] Simon P, Wilson, Francisco J S. Nonparametric Analysis of the Order-Statistic Model in Software Reliability[J]. IEEE Transactions on Software engineering, 2007, 33(3):198-208

[69] Lyu M R, He Y. Improving the N-version Programming Process through the Evolution of Design Paradigm[J]. IEEE Trans. Reliability, 1993, 42(2):179-189

[70] Wen L W, Thomas L H. A Moving Average NHPP Reliability Growth Model to Account for Software with Repair and System Structures[J]. IEEE Transactions on Reliability, 2007, 56(3):411-421

[71] 江建慧，梁剑华，靳昂，等. Linux 上软件实现的瞬时故障注入方案及实现[J]. 同济大学学报：自然科学版，2006，34(6)：823-827

[72] 邹丰忠，李传湘. 软件可靠性模型的补偿算法[J]. 数学物理学报，1999，19(1)：89-93

[73] 邹丰忠，李传湘. 软件可靠性混沌模型[J]. 计算机学报，1999，24(3)：281-291

[74] 张勇强，孙胜娟. 基于未确知理论的软件可靠性建模[J]. 软件学报，2006，17(8)：1681-1687

[75] Bai C G. Bayesian network based software reliability prediction with an operational profile[J]. The Journal of Systems and Software, 2005, 77(3):103-112

[76] Cid J E R, Achcar J A. Bayesian inference for nonhomogeneous Poisson processes in software reliability models assuming nonmonotonic intensity functions[J]. Computational Statistics and Data Analysis, 1999, 32:147-159

[77] Cai K Y. Censored software-reliability models[J]. IEEE Transactions on Reliability, 1997, 46(1):69-75

[78] Dai Y S, Xie M, Long Q, et al. Uncertainty Analysis in Software Reliability Modeling by Bayesian Approach with Maximum-entropy Principle[J]. IEEE Transactions on Software Engineering, 2007, 33(11):781-795

[79] Karunanithi N, Whitley D, Malaiya Y K. Prediction of software reliability using connectionist models[J]. IEEE Trans. Software Engineering, 1992, 18(7):563-574

[80] Karunanithi N, Whitley D, Malaiya Y K. Using Neural Networks in Reliability Prediction[J]. IEEE Software, 1992, 9(4):53-59

[81] Karunanithi N. Generalization in the Cascade-correlation Architecture: Some Experiments and Applications[D]. Computer Science Department, Colorado State University, 1992

（下转第 27 页）

las, USA, 1986: 225-232

[28] Kunii T L, Nosovskij G V, Hayashi T. A diffusion model for computer animation of diffuse ink painting[C]// Proc. of Computer Animation'95. Geneva, Switzerland, 1995: 98-102

[29] Zhang Q, Sato Y, Takahashi J, et al. Simple cellular automation-based simulation of ink behavior and its application to Suibokuga-like 3D rendering of trees[J]. Journal of Visualization and Computer Animation, 1999, 10(1): 27-37

[30] Lee J. Diffusion rendering of black ink paintings using new paper and ink models[J]. Computers and Graphics, 2001, 25(2): 295-308

[31] Way D L, Shih Z C. The synthesis of rock textures in Chinese landscape painting[C]// Proceedings of Eurographics' 2001. Manchester, 2001: 123-131

[32] 徐颂华, 徐从富, 刘智满, 等. 面向电子书画创作的虚拟毛笔模型[J]. 中国科学[E], 2004, 34(12): 1359-1374

[33] 张海江, 王秀锦, 等. 应用分形仿真水墨扩散轮廓[J]. 计算机辅助设计与图形学学报, 2004, 16(4): 554-558

[34] 齐亚峰, 孙济洲, 商毅. 中国水墨画的基本艺术特征及其计算机仿真实现[J]. 中国图象图形学报, 2003, 8(5): 562-566

[35] 石永鑫, 孙济洲, 张海江, 等. 基于粒子系统的中国水墨画仿真算法[J]. 计算机辅助设计与图形学学报, 2003, 15(6): 667-672

[36] 张海江, 王秀锦, 等. 应用分形仿真水墨扩散轮廓[J]. 计算机辅助设计与图形学学报, 2004, 16(4): 554-558

[37] 李丹, 孙美君, 孙济洲. 水墨画仿真中画笔的行为实现[J]. 中国图象图形学报, 2004, 9(02): 184-189

[38] 白海飞, 齐亚峰, 孙济洲. 基于纹理映射的非真实感"干笔飞白"效果的仿真生成[J]. 天津大学学报, 2003, 38(1): 74-79

[39] Wang Xiujin, Sun Jizhou. Simulating For Chinese Painting Based on Image Analogies[C]// Proceedings of the SPIE. 2004, 5444: 157-161

[40] 孙美君, 李丹, 孙济洲. 基于纹理合成的中国山水画系统仿真[J]. 系统仿真学报, 2004, 16(10), 2317-2320

[41] Winkenbach G, Salesin D H. Computer-generated pen-and-ink illustration[C]// Proc. of SIGGRAPH '94. 1994: 91-100

[42] Winkenbach G, Salesin D H. Rendering parametric surfaces in pen-and-ink[C]// SIGGRAPH '96 Conference Proceedings. 1996: 469-476

[43] Salisbury M P, Anderson S E, Barzel R, et al. Interactive pen-and-ink illustration[C]// ACM SIGGRAPH' 94 Conference Proceedings. 1994: 101-108

[44] Salisbury M, Wong M T, Hughes J F, et al. Orientable textures for image-based pen-and-ink illustration[C]// Proceedings of ACM SIGGRAPH 97. Los Angeles, New York, 1997: 401-406

[45] 于金辉, 徐晓刚, 彭群生. 一个三维计算机水粉笔刷模型[J]. 计算机辅助设计与图形学学报, 2000, 12(9): 664-667

[46] 张显全, 于金辉, 蒋凌琳, 等. 计算机辅助生成剪纸形象[J]. 计算机辅助设计与图形学学报, 2005, 17(6): 1378-1382

[47] 彭冬梅. 面向剪纸艺术的非物质文化遗产数字化保护技术研究[D]. 杭州: 浙江大学, 2008

[48] 钱小燕, 肖亮, 吴慧中. 一种流体艺术风格的自适应 LIC 绘制方法[J]. 计算机研究与发展, 2007, 44(9): 1588-1594

[49] 钱小燕, 肖亮, 吴慧中. 一种基于分布均匀度的非真实感蚁群绘制方法[J]. 中国图象图形学学报, 2006, 11(12): 1792-1798

[50] 钱小燕, 肖亮, 吴慧中. 一种非真实感绘制的多智能体仿真方法[J]. 系统仿真学报, 2006, 18(10): 2836-2839

[51] 辛玲, 王相海. 非真实感油画绘制的色调获取方法研究[J]. 计算机科学, 2008, 35(12): 216-219

[52] 钱小燕, 肖亮, 吴慧中. 基于多分辨率的非真实感绘制[J]. 南京理工大学学报, 2006, 30(3): 348-351

[53] Praun E, Hoppe H, Webb M, et al. Real-time hatching[C]// Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, Los Angeles, California, 2001: 581-586

[54] 方建文, 于金辉, 姚琤, 等. 利用 GPU 对 3D 角色模型绘制水墨效果[J]. 计算机辅助设计与图形学学报, 2007, 19(4): 9-12

[55] 陈为, 张海嵩, 于金辉. 基于图形硬件加速的国画场景绘制[J]. 计算机辅助设计与图形学学报, 2005, 17(11): 47-52

[56] Hertzmann A. Survey of Stroke-Based Rendering [J]. IEEE Computer Graphics & Applications, Special Issue on Non-Photorealistic Rendering, 2003, 23(4): 70-81

[57] Sloan P P, Martin W, Gooch A, et al. The lit sphere: A model for capturing NPR shading from art[C]// Graphics Inter-face. 2001: 143-150

[58] Jacobs T S. Light for the Artist[M]. Watson Guptill Publications, 1988

[59] Laning E. The Act of Drawing [M]. McGraw-Hill, New York, 1971

[60] Klein A W, Li W, MKazhdanete M. Non Photorealistie virtual environments[C]// Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. 2002: 527-534

(上接第 19 页)

[82] Su Yu-Shen, Huang Chin-Yu. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models[J]. Journal of Systems and Software, 2007, 80(4): 606-615

[83] Cai K Y, Cai L, Wang W D. On the neural network approach in software reliability modeling[J]. Journal of Systems and Software, 2001, 58(1): 47-62

[84] Aljahdali S H, Sheta A, Rine D. Prediction of software reliability: A comparison between regression and neural network non-parametric models[C]// Proc. ACM/IEEE Int. Conf. Computer Systems and Applications. Beirut, Lebanon, 2001: 470-473

[85] Aljahdali S H, Sheta A, Rine D. Predicting accumulated faults in software testing process using radial basis function network models[C]// Proc. ISCA 17th Int. Conf. Computers and Their Applications. San Francisco, 2002: 26-29

[86] Ho S L, Xie M, Goh T N. A study of the connectionist models for software reliability prediction[J]. Computers and Mathematics with Applications, 2003, 46(7): 1037-1045

[87] Sitte R. Comparison of software-reliability-growth predictions: Neural networks vs parametric-recalibration[J]. IEEE Trans. Reliability, 1999, 48(3): 285-291

[88] Tian L, Noore A. Software reliability prediction using recurrent neural network with Bayesian regularization[J]. Int. J. Neural Systems, 2004, 14(3): 165-174

[89] Tian L, Noore A. Evolutionary neural network modeling for software cumulative failure time prediction[J]. Reliability Engineering and System Safety, 2005, 87(1): 45-51

[90] Vapnik V. The Nature of Statistical Learning Theory[M]. New York: Springer Verlag, 1995

[91] Tian L, Noore A. Dynamic Software Reliability Prediction: An Approach Based on Support Vector Machines[J]. International Journal of Reliability, Quality and Safety Engineering, 2005, 12(4): 309-321

[92] Yang B, Li X. A Study on Software Reliability Prediction Based on Support Vector Machines[C]// IEEE Internal Conference on Industrial Engineering and Engineering Management. 2007: 1176-1180