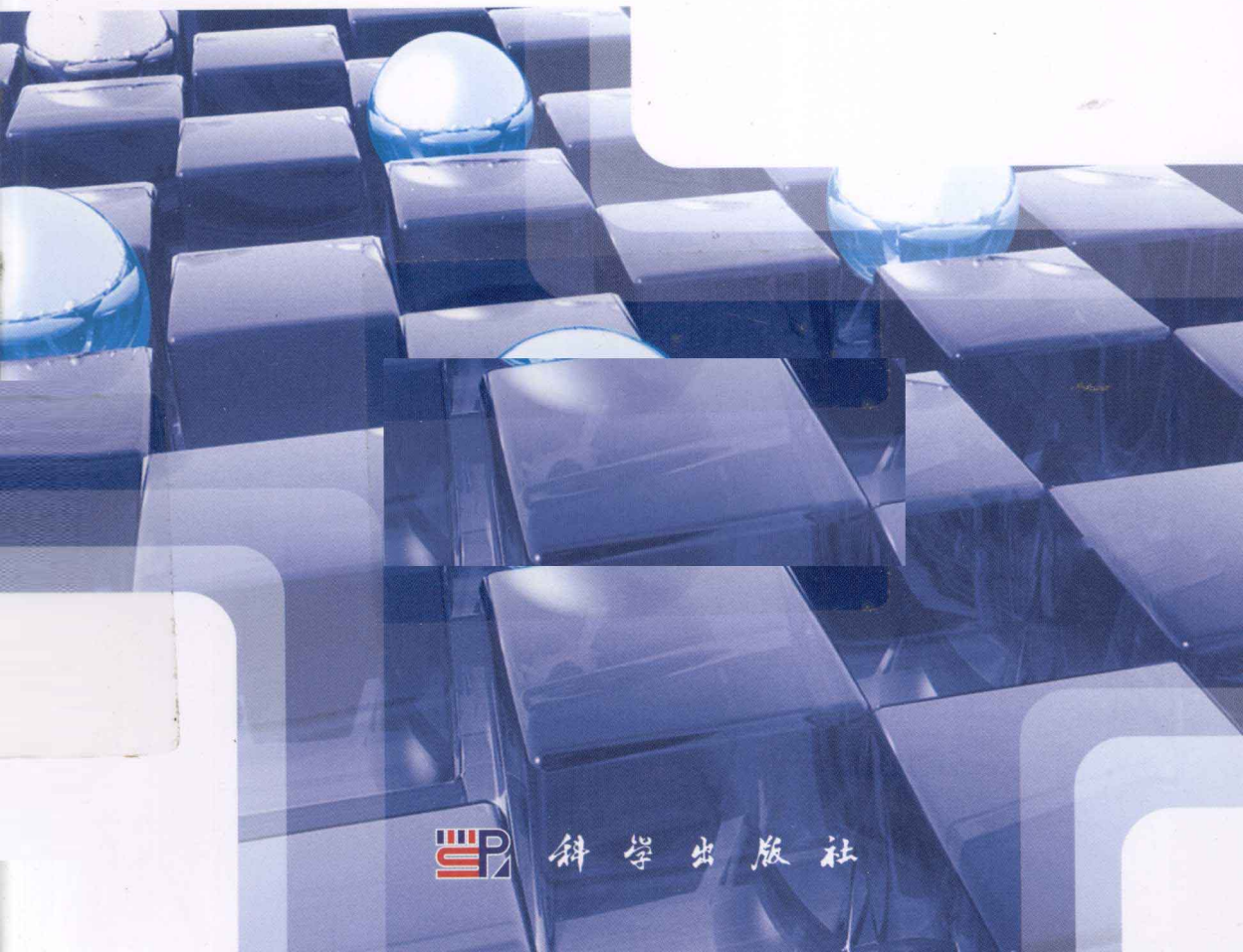
软件可靠性工程中的计算智能方法

Computational Intelligence for Software Reliability Engineering

郭 平 编著



科学出版社

内 容 简 介

软件可靠性工程是近年来计算机学科的研究热点之一，而计算智能理论和技术在其中的应用已经引起研究者的极大关注。本书结合作者近年来在该交叉领域的研究成果，对在软件可靠性工程中已经用到的以及有潜在应用价值的计算智能各个方面进行比较系统和全面的阐述。全书共 8 章。第 1 章概括介绍软件可靠性工程和计算智能。第 2～7 章介绍计算智能的主要分支，包括人工神经网络、模糊系统、演化计算（遗传算法、演化策略、演化规划、遗传程序设计、差分演化和协同演化）、群体智能方法（粒子群优化算法、蚁群优化算法、蜂群算法）、人工免疫系统、计算智能中的统计学习方法。第 8 章讨论计算智能在软件可靠性工程领域的应用。

本书可作为计算机科学与技术、信息科学与技术、电子科学与技术等专业研究生、高年级本科生教材，也可供从事软件可靠性工程、计算智能等研究和应用的相关科技人员阅读参考。

图书在版编目(CIP)数据

软件可靠性工程中的计算智能方法/郭平编著. —北京：科学出版社，2012. 5

ISBN 978-7-03-034131-0

Ⅰ. ①软… Ⅱ. ①郭… Ⅲ. ①人工神经网络—计算—应用—软件可靠性—软件工程 Ⅳ. ①TP311. 5

中国版本图书馆 CIP 数据核字(2012)第 079695 号

责任编辑：余 丁 张海丽 / 责任校对：朱光兰

责任印制：张 倩 / 封面设计 耕者设计工作室

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

骏走印刷厂印刷

科学出版社发行 各地新华书店经销

*

2012 年 5 月第 一 版 开本：B5(720×1000)

2012 年 5 月第一次印刷 印张：19 1/2

字数：379 000

定价：65.00 元

（如有印装质量问题，我社负责调换）

前　言

随着对计算机需求和依赖的与日俱增，计算机软件系统的规模和复杂性急剧增加，软件质量特别是软件可靠性问题已成为人们共同关注的焦点。在一些重要应用领域，如民航订票系统、银行结算系统、证券交易系统、自动飞行控制软件、武器装备系统、载人航天系统、军事防御和核电站安全控制系统中，对软件可靠性要求的更高。如果使用可靠性欠佳的软件，可能造成灾难性后果。因此，许多科学家在展望21世纪计算机科学发展方向和策略时，把软件质量放在优先于提高软件功能和性能的地位，因此软件可靠性工程学越来越受到产业界的高度关注。软件可靠性是软件质量的重要特性之一，软件可靠性工程是软件工程与可靠性工程的结合，是为保证经济、及时地实现软件可靠性目标而采取的系统活动和方法。

对软件可靠性工程的研究，传统方法主要建立在一般的概率论和数理统计基础上，但对于有些问题，不适合采用传统方法。软件可靠性理论和技术还有待进一步发展，需要采用新的研究方法，如近年来新发展的计算智能、模式识别等研究领域的成果。计算智能是在人工神经网络、演化计算和模糊系统三个分支发展相对成熟的基础上，通过相互之间的有机融合形成的新方法，是智能理论和技术发展的新阶段。近年来，计算智能理论与技术得到迅速的发展，在图像处理、模式识别、知识获取、经济管理、生物医学和智能控制等许多领域都得到广泛应用，取得了一系列令人鼓舞的研究成果，在软件工程方面的应用研究逐渐引起了人们的关注。

作者自从致力于研究将计算智能方法应用于软件可靠性工程以来，一直寻找对该领域有比较系统和全面论述的专著或教材，但至今仍未找到。为了进一步推动该领域的研究与应用，尤其是反映该研究领域最新的研究成果，满足广大教学和科研人员对相关资料的迫切需要，作者特编撰本书。

本书以软件可靠性工程中采用的计算智能方法为出发点，依托作者多年来在该领域的研究与实践，帮助读者更深入理解软件可靠性工程中的计算智能方法，并取得一定的应用能力。本书的目标是使读者直观快捷地理解：什么是软件可靠性工程，什么是计算智能，如何将计算智能方法应用于软件可靠性工程研究。

本书结合作者近年来在该领域的研究成果，对目前已经应用以及在软件可靠性工程中有潜在应用的计算智能各方面进行比较系统和全面的阐述和讨论。全书较系统地总结现有计算智能的理论和技术，并尽可能对软件应用领域进行分析与讨论。同时，在撰写过程中融入近年来作者在交叉领域取得的研究成果，尤其

是将计算智能应用于软件质量预测、软件可靠性模型以及模型优化等前沿问题的研究成果。

考虑到不同层次的读者——从入门者到专家级别的具有不同专业知识水平的研究者或实践者，本书各章都尽量相对独立，以适应各种需求。第 2～8 章结尾都配备思考题，既能满足多个学科不同层次读者的需要，也可以作为某一专题的单独课程、高等院校有关专业研究生的教材或参考书。在撰写过程中，作者参阅了国内外许多研究者有关软件可靠性工程及计算智能的学术论文和专著，章后附有详细的参考文献，希望这些参考文献能够帮助对某一研究方向感兴趣的读者了解自己专业领域的国际最新动态。

本书是作者及其研究团队成员多年来从事科学研究的集体智慧结晶。在撰写过程中，曾文艺教授、尹乾副教授等为本书提供了丰富的参考资料。另外，博士研究生周秀玲、张素兰、孙洋、王静、杨栋、郭歌歌、姜子恒、林松，硕士研究生邵帅、程旭超、宋梦馨等共同努力，一起协助作者完成本书的撰写工作。在此，对他们表示诚挚的感谢。本书是国家高新技术研究发展计划(863 计划)(2009AA010314)和国家自然科学基金项目(60275002、60675011、90820010)的部分研究成果总结。

限于作者水平，书中难免出现疏漏和不妥之处，有些观点和提法也可能不当，欢迎广大读者批评、指正。

郭　平

2011 年 8 月于北京

目　录

第 1 章　软件可靠性

1.1　软件可靠性工程简介

1.1.1　软件危机与软件可靠性工程

从 20 世纪 60 年代开始，计算机容量和速度的增加使得其应用范围迅速扩大，计算机软件系统的规模越来越大，复杂性越来越高，软件质量特别是软件可靠性问题也越来越突出。尽管耗费了大量的人力物力，软件质量却得不到保证，特别是由于软件可靠性差而造成的损失十分惊人，从而导致了软件危机。应用程序本身对系统运行的可靠性要求也越来越高，软件可靠性问题在一些重要应用领域尤为突出，如航空、航天领域里的民航订票系统、自动飞行控制软件、武器装备系统、载人航天系统；银行等服务性行业里的银行结算系统、证券交易系统；国防工业中的军事防御以及国民经济中的核电站安全控制系统等。即使是在日常生活一般应用程序的开发与销售过程中，市场对软件可靠性要求也越来越高。在为解决软件危机而诞生的软件工程学中，软件质量被放在优先于提高软件功能和性能的地位。软件可靠性是软件质量的重要特性之一，软件可靠性工程是一个十分实用、覆盖软件整个开发过程的软件质量保障技术[1]。

由于软件可靠性比硬件可靠性更难保证，软件当中潜在的错误会严重地影响整个系统的可靠性，因此，软件可靠性低是造成软件危机的重要原因之一。软件成为一种产品，是计算机系统的灵魂，是许多复杂系统的神经中枢和关键；但由于软件错误造成系统瘫痪、失效、人员伤亡和重大经济损失的例子也时有所闻。1963 年，美国在发射金星探测火箭的控制程序中，有一条循环语句的“,”误写为“.”，仅这一点之差，就酿成发射失败、损失达上千万美元的事故。最为典型的例子是 1996 年 6 月 4 日，欧洲宇航局发射阿里亚娜 5 号火箭升空不到 8s 就发生爆炸的惨重事故，其原因是 3 个软件问题。据美国商务部的国家标准技术研究所于 2002 年发表的有关软件缺陷的损失调查报告显示：“据推测，由于软件缺陷而引起的损失额每年估计高达 222 亿～595 亿美元，其中还不包括由于关键软件缺陷所导致的如丧失生命等灾难性后果而引起的损失。”[2] 目前，随着软件规模、复杂程度的大幅度提高，大型系统对软件依赖程度越来越严重，软件质量不高首先是因软件可靠性低而被人们广泛关注和重视，已经成为人们讨论软件危机的一个热

点。如果能够在软件企业实行软件质量管理制度的同时，广泛应用软件可靠性工程技术，这将会使国内软件企业真正能够生产出高质量的软件产品，从而参与国际软件产业的竞争。因此软件可靠性工程是一项既具有理论研究价值，又具有实际应用前景的技术。

软件可靠性工程是对软件的质量（特别是软件可靠性）进行管理和控制的实用性学科。软件可靠性工程是一项系统工程，因此在软件开发时，采取管理和技术两方面的措施都非常重要。尽管强调管理的重要性日渐突出，但在技术方面，经过近 30 多年的发展与研究，软件可靠性模型理论的研究同样取得了巨大的成就，它提高了人们对软件开发、测试过程本质的认识，对软件生命周期的各个阶段从需求分析、设计、编码、组装、测试、运行到维护都有极高的价值。软件可靠性模型理论的应用成百倍地改善了每千行代码的软件错误率。

1.1.2 软件可靠性概念

1983 年美国 IEEE 计算机学会对“软件可靠性”做出明确定义，此后该定义被美国标准化研究所接受为国家标准，1989 年我国也接受该定义为国家标准。该定义包括两方面的含义：

（1）在规定的条件下，在规定的时间内，软件不引起系统失效的概率。

（2）在规定的时间周期内，在所述条件下，程序执行所要求的功能的能力。

其中，概率是系统输入和系统使用的函数，也是软件中存在的故障的函数，系统将根据系统输入确定是否会遇到已存在的故障。

下面列出与软件可靠性相关的定义。

定义 1.1.1 软件可靠性。

软件可靠性（software reliability）是指软件在规定的运行环境中和规定的时间内无失效运行的概率[1]。它通常是时间的函数，使用 $R(t)$ 表示。

令：随机变量 t 表示发生故障的时刻，$t\in[0,\infty]$；函数 $f(t)$ 为随机变量 t 的概率密度函数，$F(t)$ 表示分布函数；$P(0\leqslant t\leqslant t_1)$ 表示从初始时刻到 t_1 时刻程序发生故障的概率。

设：初始时刻程序运行正常，即 $F(0)=0$。于是有

$$F(t)=\int_0^t f(x)\mathrm{d}x \tag{1.1.1}$$

$$f(t)=\frac{\mathrm{d}F(t)}{\mathrm{d}t} \tag{1.1.2}$$

令 $Pf(t_1)$ 表示从 0 到 t_1 时刻程序发生故障的概率，有

$$Pf(t_1)=P(0\leqslant t\leqslant t_1)=F(t_1)-F(0)=F(t_1) \tag{1.1.3}$$

如果在 $[0,t]$ 区间程序成功运行的概率为 $Ps(t)$、发生故障的概率为 $Pf(t)$，

则有

$$Ps(t)+Pf(t)=1 \tag{1.1.4}$$

式中，$Ps(t)$为可靠性，一般标记为 $R(t)$。

由以上几个公式可导出

$$R(t)=1-Pf(t)=1-F(t)=1-\int_0^t f(x)\mathrm{d}x \tag{1.1.5}$$

式(1.1.5)说明，当软件残留的错误数一定时，程序运行的时间越长，发生故障的次数越多，软件的可靠性越小[3]。

定义 1.1.2　软件故障率。

软件故障率用于描述软件发生故障的概率。

引入故障率函数 $Z(t)$，以比较一个程序在不同时期、或不同程序在同一时期的可靠性。设系统一直成功运行至时刻 t，$t\in[t_1,t_1+\Delta t]$，$P(t_1\leqslant t\leqslant t_1+\Delta t, t>t_1)$是系统在$[t_1,t_1+\Delta t]$时间间隔且 $t>t_1$ 时发生故障的概率。故障率函数 $Z(t_1)$的值定义为

$$Z(t_1)=\lim P(t_1\leqslant t\leqslant t_1+\Delta t,\ t>t_1)/\Delta t \tag{1.1.6}$$

可以证明

$$Z(t)=\frac{1}{R(t)}\cdot\frac{\mathrm{d}F(t)}{\mathrm{d}t} \tag{1.1.7}$$

式(1.1.5)的两端对时间 t 求导得

$$\mathrm{d}F(t)/\mathrm{d}t=-\mathrm{d}R(t)/\mathrm{d}t \tag{1.1.8}$$

代入式(1.1.7)，得

$$\frac{\mathrm{d}R(t)}{R(t)}=-Z(t)\mathrm{d}t \tag{1.1.9}$$

对式(1.1.9)两端积分，利用初始条件 $R(0)=1$，可得

$$R(t)=\exp\left(-\int_0^t Z(x)\mathrm{d}x\right) \tag{1.1.10}$$

式(1.1.10)即可靠性和故障率的基本方程式。据此可以导出几个常用的故障模型：

(1) $Z(t)=\lambda$，其中 λ 是常数。将 λ 代入式(1.1.10)，可得

$$R(t)=\mathrm{e}^{-\lambda t} \tag{1.1.11}$$

式(1.1.11)被称为故障率为常数的可靠性模型。由于故障率是常数，可靠性将随着时间 t 的增加呈指数衰减。

(2) $Z(t)=k_t$，这里 k 为常数，$t\geqslant 0$。将 k_t 代入式(1.1.10)，可得

$$R(t)=\mathrm{e}^{-kt^2/2} \tag{1.1.12}$$

式(1.1.12)被称为故障率为时间的线性函数的可靠性模型，即当存在损耗和退化时，故障率将随着时间的增加而线性增加。该模型一般不适合于软件产品[4]。

需要指出，软件中的错误一般都是人为的设计错误，其中多数是逻辑错误。随着对软件的测试及修复，软件系统中的错误会越来越少。因此，实际软件系统的故障率函数曲线应如图 1.1.1 所示，即软件故障率不是常数、也不是线性函数，而是按指数函数下降。实际的统计结果也说明了这一点。

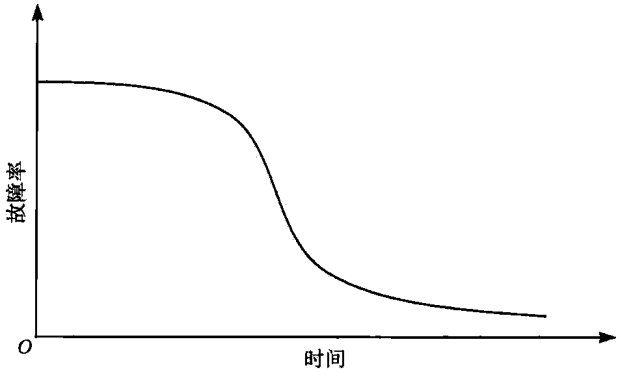


图 1.1.1 故障率函数曲线

平均无故障时间(mean time between failure，MTBF)指软件在相继两次失效之间正常工作的平均统计时间[5]。在实际使用时，MTBF 通常是指当 n 很大时，系统第 n 次失效与第 $n+1$ 次失效之间的平均统计时间。对于失效率为常数和系统恢复正常时间很短的情况下，MTBF 与平均失效前时间(mean time to failure，MTTF)几乎是相等的。MTTF 指软件在失效前正常工作的平均统计时间。

国外一般民用软件的 MTBF 为 1000h 左右。对于可靠性要求高的软件，则要求 1000～10000h。

定义 1.1.3 软件错误。

软件错误是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生[5]。可见软件错误是代码中的缺陷，是由错误引起的，是一个人或多个人的不正确或遗漏行为造成的。相对于软件本身，软件错误是一种外部行为，是一种面向开发的概念。

定义 1.1.4 软件缺陷。

软件缺陷是存在于软件(文档、数据、程序)之中的那些不希望或不可接受的偏差，如少一个逗点、多一语句等[5]。其结果是软件运行于某一特定条件时出现软件故障，这时称软件缺陷被激活。

定义 1.1.5 软件故障。

软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态[5]。譬如软件处于执行一个多余循环过程时，我们说软件出现故障。此时若无适当措施(容错)加以及时处理，便产生软件失效。显然，软件故障是一种动态行为。

定义 1.1.6　软件失效。

软件失效是指软件运行时产生的一种不希望或不可接受的外部行为结果，是系统运行行为对用户要求的偏离，是一种面向用户的概念[5]。失效意味着系统的运行，只有在执行程序的过程中才会出现软件失效。

1.1.3　软件可靠性工程的定义

1992 年，美国航空与航天学会（AIAA）在其发布的标准“ANSI/AIAA R-013-1992 推荐的软件可靠性实践”中定义软件可靠性工程是“应用统计技术处理在系统开发和运行期间所采集的数据，以便详细说明、预计、估计和评价基于软件的系统可靠性”[1]。

这个定义有两个特点：一是应用统计技术来处理有关故障数据；二是把软件可靠性工程的目的仅限于“详细说明、预计、估计和评价基于软件的系统可靠性”，以此区分软件可靠性工程和软件工程的界限。

软件可靠性工程不仅仅是用数理统计的方法来详细说明、预计、估计和评价基于软件的系统可靠性，还应该包括软件可靠性的需求、分析、设计、实施、售后及维护、工程管理活动。因此，大家认为 AIAA 发布的软件可靠性工程的定义是片面的，而软件可靠性工程的正确定义应该是：为获得软件可靠性而进行的一系列开发、维护软件产品的活动。这些活动包容了 AT&T 贝尔实验室提出的软件可靠性大纲的 20 个工作目录和 Musa 提出的软件可靠性的研究内容[6,7]。

软件可靠性工程是软件工程与可靠性工程的结合，是为保证经济、及时地实现软件可靠性目标而采取的系统的活动和方法，是软件质量的一个重要方面[1]。

1.2　软件可靠性建模

1.2.1　软件可靠性理论

自 1973 年 IEEE 计算机软件可靠性年会召开以后，软件可靠性模型理论就成为 IEEE 和其他学术机构、工业和政府部门的各种学术会议、学术著作和论文的主要研究题目之一。人们已看到了它无可限量的潜在效益，已经认识到了软件可靠性模型是软件可靠性工程的基础之一。

如何进行可靠性分析？如何建立它的结构模型和数学模型，并进行定量的评估？这是研究者目前关注的焦点之一。软件可靠性模型，对于软件可靠性估测起着核心的作用。而对于软件质量保证有直接意义的模型，是那些能够以软件的故障历史作为其参数基础来加以估计的模型。所有的模型都要求输入故障数据，但它们要求输入的数据（变量）在形式上不尽相同，一般有：操作时间、执行时间、日

历时间、排错时间、故障间隔时间、观察到的故障数、排除的错误个数等。而模型的输出依据估测目的、形式的不同也是不一样的，包括软件中的初始错误数、剩余错误数、故障密度函数值、达到特定可靠性指标(或故障率指标)所需继续测试的时间等等。软件可靠性预测的基本思想是根据软件故障发生的历史，预测软件将来故障发生的行为，软件可靠性模型应是这种思想的实现。为了保证可靠性模型的估测精度，好的软件可靠性模型应该包括对测试覆盖的说明，并且能够反映软件的错误修复过程，还应在软件生存周期的早期阶段对软件的质量进行预计，发现可能的易于出错的模块，以便软件工程师对其进行修补。

1.2.2 软件可靠性模型

1. 可靠性模型分类

软件可靠性模型是软件可靠性领域中研究最早的一个方面。依据建模对象，将软件可靠性模型分为两大类：静态模型和动态模型。

1) 静态模型

这种模型试图利用软件复杂度来确定软件缺陷数，它的建模对象是软件的各种复杂性参数。在软件工程可靠性模型中，它的重要性在于它不需要进行软件测试即可进行软件缺陷估计，这样就可以应用于软件开发的早期阶段(软件测试之前)。而随着软件可靠性设计地位的提高，这种模型在软件可靠性中的重要性也将得到提高。

静态模型中最常见的是复杂性度量模型，可以使用如下描述：

$$y = f(x_1, x_2, \cdots, x_k) + e \tag{1.2.1}$$

式中，y 为缺陷率/缺陷数目；x_i 为属性(包括软件大小、复杂度、技术层次、决策数目等)；e 为错误。

2) 动态模型

在所有的软件可靠性模型中，这一类模型是最多的。这类模型的建模对象是与运行时间有关的数据或信息，同时也可以包括与运行时间无关的数据或信息。这类模型利用测试过程中获得的软件缺陷失效时间或者在一段时间内软件失效数来计算整个软件的缺陷数和下一个软件失效发生的时间或者其他一些与软件失效有关的数据。这类模型从所使用的数学方法上，可分为马尔可夫过程模型[8]、非齐次泊松过程(NHPP)模型[9]、贝叶斯模型[10]等。

动态模型从应用的时间划分主要有两类：一类是对整个开发过程的建模；另一类是对后期测试阶段的建模。动态模型的一个通用分母是开发时间或其逻辑对应(如开发阶段)的函数。

第一类动态模型中最常见的是 Rayleigh 模型。Rayleigh 模型是 Weibull 分

布(三种著名的极值分布之一)的一员,是一个形式化的参数模型,在开发工作完成并准备发布给用户时可预测潜伏的软件缺陷[11]。

Weibull 分布的一段时间内的缺陷密度(率)或缺陷到达模式为

$$\text{PDF: } f(t)=\frac{m}{t}\left(\frac{t}{c}\right)^{m}\mathrm{e}^{-(t/c)^{m}} \tag{1.2.2}$$

Weibull 分布的累积分布方程为

$$\text{CDF: } f(t)=1-\mathrm{e}^{-(t/c)^{m}} \tag{1.2.3}$$

式中,m 为形状参数;c 为比例参数;t 为时间。一般形状参数 m 为 1 或 2 时可以应用到软件可靠性领域。其中,$m=2$ 时,为 Rayleigh 模型[12]。

第二类动态模型一般为软件可靠性增长模型。可靠性增长模型通常建立在正式测试过程的数据上。当开发完成,测试面向顾客时,出现缺陷或故障其实是用户使用时产品可靠性的良好征兆。在开发后的测试阶段,当发生故障、缺陷被确认并修复时,软件更加稳定,可靠性也随时间增长,因此这类模型一般被统称为软件可靠性增长模型[13]。指数模型是 Weibull 分布系列的特例,其形式参数 $m=1$,适合于单一衰减为渐进的统计过程,是其他软件可靠性增长模型的基础,是软件可靠性增长模型的基本形式。

2. 几种经典的软件可靠性模型

1) Jelinski-Moranda 模型

由 Jelinski 和 Moranda 在 1972 年开发的 J-M(Jelinski-Moranda)模型是最早的软件可靠性增长模型之一[14]。它包括以下的简单假设:

(1) 在测试初期,软件代码中有 u_0 个错误,u_0 是个固定数但是未知。

(2) 各个错误是相互独立的,即每个错误导致系统发生失效的可能性大致相同,各次失效的间隔时间也相互独立。

(3) 不管错误什么时候发生,发生时立刻被排除,而且不会给软件带来新的错误。

(4) 程序的失效率在每个失效间隔时间内是常数,其数值正比于程序中残留的错误数,在第 i 个测试区间,其失效率函数为

$$\lambda(x_i)=\varphi(u_0-i+1) \tag{1.2.4}$$

式中,φ 为比例常数;x_i 为第 i 次失效间隔中以 $i-1$ 次失效为起点的时间变量。

(5) 程序测试环境与预期的使用环境相同。

在这些基本假设的基础上,J-M 模型认为,以第 $i-1$ 次失效为起点的第 i 次失效发生的时间是一个随机变数,它服从以 $\varphi(u_0-i+1)$ 为参数的指数分布,其密度函数为

$$f(x_i)=\varphi(u_0-i+1)\exp(-\varphi(u_0-i+1)x_i) \tag{1.2.5}$$