第十一章 软 件 可 靠 性

§11-3 软件可靠性的数学模型

一、软件可靠性建模概述

二、J - M (Jelinski—Moranda) 模型

三、Halstead模型

四、G－O (Goel一Okumoto) NHPP模型

§11-3 软件可靠性的数学模型

一、软件可靠性建模概述

软件可靠性建模旨在根据与软件可靠性(故障)有关的数据以统计方法给出软件可靠性的估计值或预测值，是从本质上理解软件可靠性行为的关键之一。

软件可靠性建模活动最早可追溯到Hudson的工作，他当时以随机生灭过程描述软件缺陷的引入和剔除过程，并证明被剔除的缺陷数服从二项式分布，其均值时间函数具有威布尔(Weibull)分布形式。

1. 模型种类

软件可靠性建模研究真正获得重视始于60年代末“软件危机”的提出，并在70年代获得巨大的发展。

现已提出了数十种模型，主要的软件可靠性模型有以下**8种**模型：

(1) Jelinski-Moranda模型

Jelinski和Moranda于1972年正式出版有关他们的模型的论著。这一模型是软件可靠性系统研究的真正开端，至今在软件可靠性建模方面仍具有重要意义，这是因为：

① 这一模型首次将硬件可靠性的基本概念，如可靠度、故障强度(率)等，系统地引入软件可靠性领域；

② 这一模型确立了软件可靠性建模的黑箱方法，即不涉及软件内部结构，仅依据软件外部行为(输入输出关系)刻画软件可靠性行为；

③ 这一模型给出了软件可靠性建模的两个主要假设：一是测试用例的选取代表软件实际运行剖面；二是不同软件故障独立发生；

④ 这一模型简单，众多的后续模型可视为这一模型的某种变形。

(2) Halstead模型

Halstead依据其“软件科学”的思想于1972年提出此模型，试图利用软件复杂度确定软件缺陷数。

此模型的重要性在于它可应用于软件开发的早期阶段（软件测试之前)。随着软件可靠性设计地位的提高，此模型的重要性也将显著增长。

(3) Littlewood-Verrall模型

Littlewood和Verrall于1973年发表此模型，其主导思想是将软件故障强度视为随机变量，从而开创了软件可靠性建模的Bayes方法。

此模型也间接考虑了发现的缺陷不被完全剔除的可能性。

(4) Musa执行时间模型

Musa于1975年发表此模型，主要贡献在于提出将**CPU执行时间作为软件可靠性的时间基准**。

(5) Sukert模型比较工作

Sukert于1976年给出其关于不同模型的分析比较的试验报告，开创了软件可靠性模型比较研究的先河。

模型比较研究已成为建模研究的一个基本方面。

(6) Nelson模型

Nelson最早于1973年提出一个基于数据域模型并于1978年得以完善。

此模型是基于数据域模型的代表，成为目前应用最多的模型之一。

(7) Goel—Okumoto NHPP模型

GoeI和Okumoto于1978年发表此模型，将软件故障次数描述为非齐次Poisson过程。此模型是NHPP模型的代表。

(8) Musa软件可靠性数据

Musa于1979年发表了一批软件可靠性数据，对软件可靠性建模研究产生重要影响。目前已成为软件可靠性建模研究的“标准”参考数据。之后Musa,Ianninohe和Okumoto于1987年又发表了“软件可靠性”一书，在软件可靠性领域颇受推崇。该书首次对众多软件可靠性模型给出了统一的数学描述。

2.模型组成

软件可靠性模型通常由以下几个部分组成：

(1)模型假设

模型是实际情况的简化或规范化，总要包含若干假设。

① 代表性假设

此假设认为软件测试用例的选取代表软件实际的运行剖面，甚至认为测试用例是独立随机地选取。

此假设实质上是指可以用测试产生的软件可靠性数据预测运行阶段的软件可靠性行为。

② 独立性假设。

此假设认为软件故障是独立发生于不同时刻，一个软件故障的发生不影响另一个软件故障的发生。

譬如在概率范畴内，假设相邻软件故障间隔构成一组独立随机变量，或假设一定时间内软件故障次数构成一个独立增量过程。

③ 相同性假设。

此假设认为所有软件故障的后果(等级)相同，即建模过程只考虑软件故障的具体发生时刻，不区分软件故障的性质。

(2) 性能度量

软件可靠性模型的输出量变为性能度量，如故障强度、残留缺陷数、可靠度等。在软件可靠性模型中性能度量通常以数学表达式给山。

(3)参数估计方法

某些可靠性度量的实际值无法直接获得(如残留缺陷数)，这时需通过一定的方法估计参数的值，从而间接确定可靠性度量的值。

当然，对于可直接获得实际值的可靠性度量，便无需参数估计了。

(4)数据要求

一个软件可靠性模型要求一定的输入数据，即软件可靠性数据。

不同类型的软件可靠性模型可能要求不同类型的软件可靠性数据。

本书仅介绍三种最常用的软件模型。

二、J - M (Jelinski—Moranda) 模型

J - M模型是最早的软件可靠性马尔科夫过程的数学模型，沿用了硬件可靠性的方法。

J – M的假设是：

(1) 软件最初的错误数为常值N。用$t_i(i=1，2，\ldots，N)$表示在第$(i-1)$个错误被改正后，软件投入运行至第i次故障发生前这个阶段的时间，且故障间隔时间t_i是统计独立的。

(2) 在两个相继故障构成的时间间隔内，软件的故障率为常数，其大小正比于软件中的残存错误数。

(3) 每个软件错误一经发现立即被修正，且不考虑修改错误的时间，也不产生新的错误。

(4)每次故障后总有并仅有一个错误被排除，因此系统的故障率是阶梯式下降的。

1.可靠性特征量

(1) 按上述假设得出的软件系统故障的密度函数为：

$$f(t_i)=\lambda_i e^{-\lambda_i t_i} \qquad i=1,2,...N \tag{11-1}$$

(2) 故障率为：

$$\lambda_i=\phi[N-(i-1)] \tag{11-2}$$

式中 ϕ 是比例常数，与 λ_i 递减率无关。

(3) 可靠度函数为：

$$R_i(t) = e^{-\phi[N-(i-1)]t} \qquad (11\text{-}3)$$

2. 可靠性特征量估计

模型的待估参数为 ϕ 和 N。设 t_1，t_2，…，t_n 是试验中实际测得的 n 次故障发生的时间间隔，则其极大似然方程为：

$$L\pi = \prod_{i=1}^{n} f(t_i)$$

$$= \prod_{i=1}^{n} \phi\left[N-(i-1)\right] e^{-\phi\left[N-(i-1)\right]t_i} \qquad (11-4)$$

运用极大似然估计法，则

$$\begin{cases} \dfrac{\partial \ln L\pi}{\partial \phi} = \dfrac{n}{\phi} - \sum_{i=1}^{n} [N-(i-1)]t_i = 0 \\ \dfrac{\partial \ln L\pi}{\partial N} = \sum_{i=1}^{n} \dfrac{1}{N-(i-1)} - \sum_{i=1}^{n} \phi t_i = 0 \end{cases} \quad (11-5)$$

可得模型参数的极大似然估计值 $\hat{\phi}$ 和 $\hat{N}$ 分别为：

$$\begin{cases} \hat{\phi} = \dfrac{n}{\hat{N}\sum_{i=1}^{n} t_i - \sum_{i=1}^{n}(i-1)t_i} \\ \sum_{i=1}^{n} \dfrac{1}{\hat{N}-(i-1)} = \dfrac{n}{\hat{N} - [\sum_{i=1}^{n}(i-1)t_i]/\sum_{i=1}^{n} t_i} \end{cases} \qquad (11-6)$$

发现 n 个软件错误后，停止测试。此时，软件的可靠性水平为：

(1) 残存错误数

$$N_T = \hat{N} - n \qquad (11-7)$$

(2) 故障率

$$\lambda_T = \hat{\phi}[\hat{N} - n] \qquad (11-8)$$

(3) 可靠度

$$R_i(t) = e^{-\hat{\phi}[\hat{N}-n]t} \qquad (11-9)$$

(4)平均无故障工作时间MTBF

$$MTBF = \frac{1}{\hat{\phi}(\hat{N} - n)} \qquad (11-10)$$

例11-1 Jelinski和Moranda曾在美国海军舰队计算机程序编制中心利用海军战术数据系统(NTDS)的软件故障数据，对他们的模型作了验证。NTDS包括38个不同的模块，每个模块都经过开发、测试、使用三个阶段。验证所用的故障数据是取自二个大型模块的软件反常报告。这个模块在设计阶段共发现了26个错误，测试阶段又发现了5个（表11-1中34-31）错误，在使用阶段又发现了3个（表11-1中31-26）错误。NTDS的数据见表11-1，J-M根据交付使用前的数据计算的结果也示于表11-1。

表11-1 NTDS数据及预计结果

阶段	错误数 n	错误间隔时间(日)	累计时间(日)	J．M预计的故障时间
设计阶段	1	9	9	4.7
	2	12	21	4.8
	3	11	32	5.9
	4	4	36	5.2
	5	7	43	5.4
	6	2	45	5.6
	7	5	50	5.8
	8	8	58	6.0
	9	5	63	6.3
	10	7	70	6.6
	11	1	71	6.9
	12	6	77	7.2
	13	1	78	7.6
	14	9	87	8.0
	15	4	91	8.5
	16	1	92	9.0
	17	3	95	9.6
	18	8	98	10.3
	19	6	104	11.1
	20	1	105	12.0
	21	1	116	13.0
	22	33	149	14.3
	23	7	156	15.9
	24	91	247	17.8
	25	2	249	20.3
	26	1	250	23.5

续表11-1 NTDS数据及预计结果

测试阶段	27 28 29 30 31	87 47 12 9 135	337 384 396 405 540	28.1 34.8 45.6 66.4 121.7
使用阶段	32 33 34		798 814 849	总的错误数 $\hat{N}$ = 31.2 故障率系数 $\hat{\phi} = 0.00685$

将表中数据代入式(11-6)得：

$$\hat{N} = 31.2158 \approx 31.2,\ \hat{\phi} = 0.00685$$

估计软件错误为31.285个,与实际大模块34个错误，较为符合。

三、Halstead模型

1．模型类别：静态经验模型。

2．模型假设

(1) 程序是良好结构化的，即程序不包含操作符、操作数、表达式的问题。

(2) 以下方程近似成立：

$$L = l_1 \log_2 l_1 + l_2 \log_2 l_2 \qquad (11-11)$$

式中：l_1 —— 为程序中不同操作符个数；

l_2 —— 为程序中不同操作数个数。

$$L = L_1 + L_2$$

其中：L_1为程序中操作符出现总次数；

L_2为程序中操作数出现总次数。

3．参数估计

程序残留缺陷 N 的估计值为：

$$\hat{N} = V / E_0 \qquad (11-12)$$

这里V表示程序容量， 即 $V = L \log_2(l_1 + l_2)$

而E_0为一常数 ($E_0 \approx 3000$)。

例11-2 某一汇编语言程序，由9个模块组成，包含约24955条汇编语言指令，如表11-2所列。

表11-2 某汇编程序原始数据及计算结果

模块	指令条数	缺陷数	决策点数	调用次数	l_1	l_2	$\hat{N}$
MA	4 032	102	372	283	47l	442	102
MB	1 329	81	215	44	180	176	18
MC	5 453	93	552	362	610	574	93
MD	l 674	26	11l	130	231	20l	26
ME	2 05l	71	315	197	366	138	71
MF	2 513	37	217	186	322	287	37
MG	699	16	104	32	131	76	16
MH	3 792	50	233	110	252	603	50
MX	3 412	80	416	230	433	357	80

解：

假设每条指令包含一个操作符和一个操作数，则　　$L = L_1 + L_2 = 2 \times$ 指令条数

又设l_1=决策点数 / 3 + 调用次数 + 64。

其中，64表示总共有64种机器语言指令。那么l_2可由式(11-11) 计算，代入式(11-12) 可以得到程序残留缺陷的估计值 $\hat{N}$，见表11-2中最后一列中的值。

由以上分析可以看出，Halstead模型认为程序中的操作符和操作数与其残留缺陷密切相关。该模型的优点是无需测试数据。

因此，优点可用于测试之前的软件开发阶段；该模型的缺点是其有效性有待进一步确认。

尽管众多事实表明，V / N可近似为常数，但E_0取值却未必为3000。也许利用程序的若干模块缺陷数(据此可估计E_0)去估计其它模块的缺陷数更为合理。

四、G-O (Goel-Okumoto) NHPP模型

1. 模型类别

模型为 动态、宏观、面向错误数模型。

2. 模型假设

(1) 软件(测试)运行方式与预计(实际)运行剖面相同。

(2) 对于任一组有限时间点 $t_1<t_2<\ldots<t_n$，在对应时间段$(0，t_1)$，$(t_1，t_2)$，…，$(t_{m-1}，t_m)$内分别发生的错误数为f_1，f_2，…，f_m相互独立。

(3) 每一缺陷有均等机会被检测发现，且等级相同。

(4) 任一时刻 t 发生的软件故障累积次数N (t)服从均值为$m(t)$的**Poisson分布**，均值$m(t)$使得微小时间段$(t, t+\Delta t)$内软件故障发生次数与 t 时刻软件残留缺陷数成正比。

$$m(t+\Delta t)-m(t) = b\left[a-m(f)\right]\Delta t \qquad (11-13)$$

式中 a — 为测试过程最终软件发生故障的总数；

b — 为比例常数。

(5) $m(t)$是一个有界、非减函数，且

$$\begin{cases} m(t) = 0 & t = 0 \\ m(t) = \alpha & t \to \infty \end{cases} \qquad (11-14)$$

由假设(3) ～ (5)可知：

$$\frac{dm(t)}{dt} = ab - bm(t)$$

$$\to m(t) = a(1 - e^{-bt}) \qquad (11-15)$$

由于累积故障数$N(t)$是均值为 $m(t)$ 的Poisson分布，那么

$$P\{N(t)=n\} = \frac{[a(1-e^{-bt})]}{n!} e^{-a(1-e^{-bt})} \quad \geq 0$$

(11 – 16)

$$P\{N(\infty)=n\} = \frac{a^n}{n!} e^{-a}$$

(11-17)

3．参数估计

设(t_{i-1}, t_i)时间内发生的软件故障次数为$N(t_i)-N(t_{i-1})$,其概率分布为：

$$P\{N(t_i)-N(t_{i-1})=k_i\}$$

$$=\frac{[m(t_i)-m(t_{i-1})]^{k_i}\exp[m(t_{i-1})-m(t_i)]}{k_i!}$$

(11 – 18)

似然函数为：

$$L\pi = \prod_{i=1}^{m} \frac{\left[m(t_i) - m(t_{i-1})\right]^{k_i} \exp\left[m(t_{i-1}) - m(t_i)\right]}{k_i!}$$

$$= \prod_{i=1}^{m} \frac{\left[a(e^{-bt_{i-1}} - e^{-bt_i})\right]^{k_i} \exp\left\{a(e^{-bt_i} - e^{-bt_{i-1}})\right\}}{k_i!}$$

(11 – 19)

参数 a，b 的估计值可由以下方程组决定：

$$\begin{cases} \hat{a} = \dfrac{\sum_{i=1}^{m} k_i}{1 - e^{-\hat{b}t_m}} \\ \dfrac{t_m e^{-\hat{b}t_m} \sum_{i=1}^{m} k_i}{1 - e^{-\hat{b}t_m}} = \sum \dfrac{\left[t_i e^{-\hat{b}t_i} - t_{i-1} e^{-\hat{b}t_{i-1}} \right]}{e^{-\hat{b}t_{i-1}} - e^{-\hat{b}t_i}} \end{cases}$$

$$(11-20)$$

4．可靠性指标

(1)故障强度

软件故障强度表示单位时间内软件发生故障的概率。即：

$$\lambda(t)=\frac{dm(t)}{dt}=\hat{a}e^{-\hat{b}t} \qquad (11-21)$$

(2) t 时刻软件剩余的故障数

t时刻软件剩余的故障(尚未发生的故障)数为：$\overline{N}(t)=N(\infty)-N(t)$

$\overline{N}(t)$ 表示 (t,∞) 时间内软件故障次数。

有假设（2）得：

$$P\left\{\bar{N}(t)=x\right\}=\frac{\left[m(\infty)-m(t)\right]^{x}}{x!}e^{-m(t)}$$

$$=\frac{\left(e^{-bt}\right)}{x!}a(1-e^{-bt}) \qquad (11\text{-}22)$$

(3) 可靠度

软件第 n 次故障发生在s时刻，发生第 n 次故障后可靠度函数为：

$$
\begin{aligned}
&R_{n+1}(x|t_n = s) = P\left[x_{n+1} > x \middle| T_n = s\right] \\
&= \exp\left[-a(e^{-bs} - e^{-b(s+x)})\right] \\
&= \exp\left[-m(x)e^{-bs}\right]
\end{aligned}
\qquad (11\text{-}23)
$$

(4) 平均失效前时间(MTTF)

给定软件第 n 次故障发生在 s 时刻，则此时软件MTTF为：

$$\begin{aligned}\mathrm{MTTF}_{n+1} &= \int_0^{\infty} R_{n+1}(x|T_n = s)dx \\ &= \int_0^{\infty} \exp\left[-m(x)e^{-bs}\right]dx \\ &\neq \frac{1}{\lambda(s)}\end{aligned} \qquad (11-24)$$

中国可靠性网

http://www.kekaoxing.com

感谢 kingdodoo 分享