第5章

软件可靠性增长模型


第5章视频

本章学习目标

- 熟练掌握经典软件可靠性增长模型建模方法。
- 了解NHHP类软件可靠性增长模型。

本章先介绍4种经典软件可靠性增长模型建模方法，然后介绍NHHP类软件可靠性增长模型以及统一的SRGM框架模型。

5.1 经典软件可靠性增长模型

随着计算机和信息处理的广泛应用，计算机系统的软件可靠性问题越来越得到人们的关注。而复杂装备系统软件规模日益增大，复杂性日益增强，使其软件可靠性问题变得更为突出。所谓软件可靠性是指软件系统在规定环境下和给定时间内无故障运行的概率，是软件质量的一个重要组成部分。软件可靠性模型是软件可靠性评估与预测的核心。目前公开发表的软件可靠性模型已经有一百多种，但是由于它们假设不同、性能各异，因此，即使用它们估测同一软件，也可能得到存在巨大差异的预测结果。至今仍然没有一个既简单又广泛适用的软件可靠性模型。

准确建立软件可靠性模型并预测其可能的增长趋势，对于确定整个软件产品的可靠性至关重要。软件可靠性增长模型（Software Reliability Growth Model, SRGM）提供了与软件代码紧密相关的可靠性特征信息，例如剩余故障数量、累积检测或排除的故障数量等。SRGM在度量、预测、提高与保证软件可靠性上被广泛应用，同时，定量地对涵盖测试资源与成本管控和最优发布时间抉择等在内的软件开发活动提供重要决策支持，是可靠性工程研究领域的重要手段。

由于软件可靠性更多、更直接地源于开发人员在设计与编制程序过程中所带来的内在错误（error），在测试阶段中，大量测试计划的有效实施可促使这些错误引发故障（fault），当达到一定条件时，故障会引发系统失效（failure）。这样，SRGM通过对失效的检测能够促使开发人员回溯至代码的层面进行调试修改与排错，进而提高可靠性。同时，SRGM针对一段时间内的失效与排除故障数量可揭示出当前测试环境下可靠性随时间变动的规律。因此，多年来，研究人员在SRGM上积累的技术框架在可靠性过程综合管理上可有效确保软件可靠性得到持续提高，尤其是在测试阶段。同时，SRGM在测试资源分配、成本管控、发布策略上得到了

应用上的广泛认可。

SRGM对软件可靠性研究形成了重要的理论支撑，对有效解决软件可靠性动态定量增长问题开辟了途径。在具体研究软件可靠性增长的突破点与线索上，SRGM以失效的观察和记录、故障的检测和排除为主线，采用微分建模的方法描述（累积检测/修复的）故障个数与软件可靠性间的数学关联，并据此指导测试策略的优化执行，以检测与排除更多的故障，提高软件可靠性。

5.1.1 JM模型

JM模型是由Z. Jelinski和P. Moranda于1972年提出的可靠性数学模型，该模型以一种简便和合乎直觉的方式表明如何根据软件缺陷的检测历程预计未来软件可靠性的行为。它包含了软件可靠性建模的若干典型和主要的假设，是最具代表性的早期软件可靠性马尔可夫过程的数学模型。随后的许多工作都是在它的基础上加以改进而提出的，因此，在这个意义上，JM模型可以视为软件可靠性研究领域的第一个里程碑。

1. 模型假设

JM模型有以下假设：

（1）软件中固有的初始错误个数N_0为一个未知常数。

（2）软件中的各个错误是相互独立的，每个错误导致系统发生失效的可能性大致相同，各次失效时间间隔也是相互独立的。

（3）测试过程中错误一旦被检测出即被完全排除，每次排错只排除一个错误，排错时间可以忽略不计，在排错过程中不引入新的错误。

（4）软件的失效率在每个失效间隔时间内是常数，其数值正比于软件中残留的错误数，比例系数为ϕ。

（5）软件的运行方式与预期的运行方式相似。

2. 模型形式

在给定N_0和ϕ的条件下，若记X_i表示第$i-1$次故障时刻到第i次故障时刻间的时间，假定故障时间间隔$X_1, X_2, \cdots, X_n$是相互独立的服从指数分布的随机变量。根据假设，在第一个错误被排除之后，失效率就由$N_0\phi$变为$(N_0-1)\phi$，以此类推，则在第i个测试区间，其失效率函数为

$$z(x_i) = \phi[N_0 - (i-1)] \tag{5.1}$$

其中，x_i为时间间隔X_i的取值。

在整个测试过程中，软件的失效率变化曲线如图5.1所示。

由假设知，在每两个错误之间只有唯一的失效率，则关于X_i的概率密度为

$$f(x_i) = \phi[N_0 - (i-1)] \exp\{-\phi[N_0 - (i-1)]x_i\} \tag{5.2}$$

其分布函数为

$$F(x_i) = \int_0^{x_i} f(x_i)\mathrm{d}x_i = 1 - \exp\{-\phi[N_0 - (i-1)]x_i\} \tag{5.3}$$

其可靠度函数为

$$R(x_i) = 1 - F(x_i) = \exp\{-\phi[N_0 - (i-1)]x_i\} \tag{5.4}$$

其平均失效时间间隔为

$$\mathrm{MTBF} = E[X_i|X_1, X_2, \cdots, X_{i-1}] = \int_0^{\infty} R(x_i)\mathrm{d}x_i = \frac{1}{\phi(N_0 - (i-1))} \tag{5.5}$$

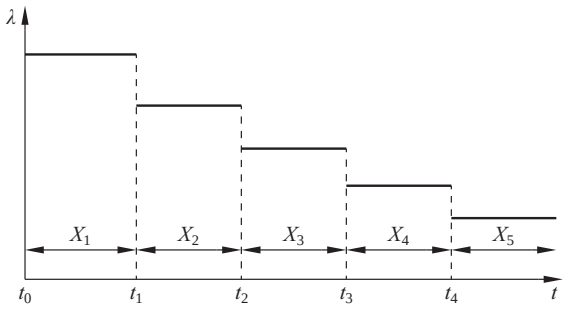

图 5.1 软件的失效率变化曲线

3. 参数估计

假定总共发生了 n 次失效，似然函数为

$$L(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} f(x_i) = \prod_{i=1}^{n} \phi[N_0 - (i-1)] \exp\{-\phi[N_0 - (i-1)]x_i\} \tag{5.6}$$

对式 (5.6) 两边取对数，有

$$\begin{aligned} \ln L(x_1, x_2, \cdots, x_n) &= \sum_{i=1}^{n} \ln f(x_i) \\ &= \sum_{i=1}^{n} \{\ln(\phi[N_0 - (i-1)]) - \phi[N_0 - (i-1)]x_i\} \end{aligned} \tag{5.7}$$

则模型参数的最大似然估计值是以下方程组的解：

$$\begin{cases} \hat{\phi} = \dfrac{n}{\hat{N}_0(\sum\limits_{i=1}^{n} x_i) - \sum\limits_{i=1}^{n}(i-1)x_i} \\ \sum\limits_{i=1}^{n} \dfrac{1}{\hat{N}_0 - (i-1)} = \dfrac{n}{\hat{N}_0 - \left(1/\sum\limits_{i=1}^{n} x_i\right)\left(\sum\limits_{i=1}^{n}(i-1)x_i\right)} \end{cases} \tag{5.8}$$

对式 (5.8) 进行求解，则可以得出 $\hat{N}_0$、$\hat{\phi}$ 的值。

4. 应用案例

这里以美国海军战术数据系统（Naval Tactical Data System, NTDS）数据集为例，其数据集形式如表5.1 所示。其中，前 26 个错误是在开发阶段发现的，第 27~31 个错误是在测试阶段发现的，在使用阶段发现了第 32 个错误，最后在再测试阶段发现了第 33、34 个错误。

将表5.1中的 n 和 x_i 代入式 (5.8) 中可得 $\hat{N}_0 = 31.7734 \approx 32, \hat{\phi}$=0.006 68，即估计出软件共有 32 个错误。实际上，此软件各个阶段共有 34 个错误，可见计算结果与实际数据大致

吻合。

表 5.1 NTDS 数据集形式

错误数	错误间隔时间	错误数	错误间隔时间	错误数	错误间隔时间
1	9	13	1	25	2
2	12	14	9	26	1
3	11	15	4	27	87
4	4	16	1	28	7
5	7	17	3	29	12
6	2	18	3	30	9
7	5	19	6	31	135
8	8	20	1	32	258
9	5	21	11	33	16
10	7	22	33	34	35
11	1	23	7		
12	6	24	91		

5.1.2 GO 模型

GO 模型是由 Goel 和 Okumoto 于 1979 年提出的，首次采用了非齐次泊松过程（NHPP）刻画软件的失效过程，属于 NHPP 有限错误模型，GO 模型也被称为软件错误查出的指数类增长模型。此后有很多新的模型是以它为基础提出的。

1. 模型假设

GO 模型有以下假设：

（1）程序在同实际执行环境相差不大的条件下执行。

（2）在软件测试过程中，所有检测到的故障是相互独立的。

（3）测试未运行时的软件失效次数为 0；当测试进行时，软件失效次数服从均值为 $m(t)$ 的非齐次泊松过程。

（4）t 时刻累积检测到的故障数量 $m(t)$ 变化率与当前剩余的故障数量 $a-m(t)$ 成正比例，比例系数为 b。

（5）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。

2. 模型形式

设 $N(t)$ 和 $m(t)$ 分别表示区间 $[0,t]$ 内的累积错误数和期望错误数，a 为软件中的初始故障数，b 为在 t 时刻每个错误的检出率，$b>0$，且有 $m(0)=0, m(+\infty)=a$。在 $[t,t+\Delta t]$ 内期望的错误发生数与时刻 t 时期望的剩余错误数成比例，于是有

$$m(t+\Delta t)-m(t)=b(a-m(t))\Delta t \tag{5.9}$$

令 $\Delta t \to 0$，则有

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b(a-m(t)) \tag{5.10}$$

利用边界条件可得GO模型的均值函数表达式：

$$m(t)=a(1-\mathrm{e}^{-bt}),\quad a>0,b>0 \tag{5.11}$$

易知GO模型的失效强度函数为

$$\lambda(t)=\frac{\mathrm{d}m(t)}{\mathrm{d}t}=ab\mathrm{e}^{-bt} \tag{5.12}$$

随机变量序列$\{X_n,n=1,2,3,\cdots\}$表示故障间隔时间，则$S_n=\sum_{k=1}^{n}X_k\ (n=1,2,3,\cdots)$表示第$n$个故障出现的时间，在给定$S_{n-1}=s$的条件下，$X_n$的条件可靠性函数（即软件可靠度）为

$$\begin{aligned}R_{X_n|S_{n-1}}(t|s)&=P(S_n-S_{n-1}>t|S_1,S_2,\cdots,S_{n-1})\\&=\exp\{-[m(t+S_{n-1})-m(S_n)]\}\\&=\exp\{-a[\mathrm{e}^{-bs}-\mathrm{e}^{-b(t+s)}]\}\end{aligned} \tag{5.13}$$

3. 模型参数估计

1）利用完全数据（故障间隔时间）估计模型参数

假设在软件测试过程中检测到的N个故障的时间$S=(S_1,S_2,\cdots,S_N)$满足$0\leqslant S_1\leqslant S_2\leqslant\cdots\leqslant S_N$，则$S$的概率密度函数是在给定$S$的观测值$s=(s_1,s_2,\cdots,s_N)$时关于$a$、$b$的似然函数：

$$L(s_1,s_2,\cdots,s_N;a,b)=\left(\prod_{k=1}^{N}ab\mathrm{e}^{-bs_k}\right)\exp[-a(1-\mathrm{e}^{-bs_N})] \tag{5.14}$$

对于给定故障发生时间的$s=(s_1,s_2,\cdots,s_N)$，令

$$\frac{\partial\ln L(s_1,s_2,\cdots,s_N;a,b)}{\partial a}=\frac{\partial\ln L(s_1,s_2,\cdots,s_N;a,b)}{\partial b}=0 \tag{5.15}$$

则有

$$\begin{cases}\dfrac{N}{\hat{a}}=1-\mathrm{e}^{-\hat{b}s_N}\\[2ex]\dfrac{N}{\hat{b}}=\displaystyle\sum_{k=1}^{N}s_k+\hat{a}s_N\mathrm{e}^{-\hat{b}s_N}\end{cases} \tag{5.16}$$

对式(5.16)进行求解，则可以得出$\hat{a}$、$\hat{b}$的值。

2）利用不完全数据（累积故障数）估计模型参数

设在时刻$t_k(k=1,2,\cdots,n)$查出的累积故障数为y_k，则关于a、b的似然函数为

$$L=P\{N(t_1)=y_1,N(t_2)=y_2,\cdots,N(t_n)=y_n\} \tag{5.17}$$

$$=\prod_{k=1}^{n}\frac{[a(\mathrm{e}^{-bt_{k-1}}-\mathrm{e}^{-bt_k})]^{y_k-y_{k-1}}}{(y_k-y_{k-1})!}\exp\left(-a(1-\mathrm{e}^{-bt_n})\right) \tag{5.18}$$

其中$t_0=0,y_0=0$。令$\dfrac{\partial\ln L}{\partial a}=\dfrac{\partial\ln L}{\partial b}=0$，则有

$$\begin{cases}\hat{a}(1-\mathrm{e}^{-\hat{b}t_n})=y_n\\[2ex]\hat{a}t_n\mathrm{e}^{-\hat{b}t_n}=\displaystyle\prod_{k=1}^{n}\frac{(y_k-y_{k-1})(t_k\mathrm{e}^{-\hat{b}t_k}-t_{k-1}\mathrm{e}^{-\hat{b}t_{k-1}})}{\mathrm{e}^{-\hat{b}t_k}-\mathrm{e}^{-\hat{b}t_{k-1}}}\end{cases} \tag{5.19}$$

对式(5.19)进行求解，则可以得出$\hat{a}$、$\hat{b}$的值。

4. 应用案例

选取美国海军战术数据系统收集的故障数据集DS1，如表5.2所示。

表 5.2 DS1 数据集

失效序号	失效时间	失效序号	失效时间	失效序号	失效时间	失效序号	失效时间
0	0	9	63	18	98	27	337
1	9	10	70	19	104	28	384
2	21	11	71	20	105	29	396
3	32	12	77	21	116	30	405
4	36	13	78	22	149	31	540
5	43	14	87	23	156	32	798
6	45	15	91	24	247	33	814
7	50	16	92	25	249	34	849
8	58	17	95	26	250		

对于DS1取前28个数据点拟合参数，后6个数据点用于验证模型，采用最大似然估计法估计出模型参数，得到两个参数的估计值：$\hat{a}=33.9935,\hat{b}=0.005\ 79$。

5.1.3 MO模型

MO模型也称对数泊松执行时间模型，是另一类被广泛使用的软件可靠性增长模型，它是由Musa和Okumoto提出的。

MO模型把排错过程中的累积失效数作为时间函数，将其近似为一个非齐次泊松过程。该模型定义了一个随机过程$\{N(t),t\geqslant 0\}$，它表示时刻t的累积失效数，其均值函数为$m(t)=E[N(t)]$，其失效强度函数为$\lambda(t)=\mathrm{d}m(t)/\mathrm{d}t$。

1. 模型假设

MO模型有以下假设：

（1）当$t=0,N(0)=0$，即测试开始前，无失效发生。

（2）失效强度随着失效期望数的增加而呈指数递减，即$\lambda(t)=\lambda_0\mathrm{e}^{-\theta m(t)}$。其中，$\lambda_0>0$是初始失效强度，$\theta>0$是失效强度递减幅度参数，$m(t)$为均值函数。

（3）当Δt足够小时，时间区间$(t,t+\Delta t)$内发生一次及以上失效的可能性为$\lambda(t)\Delta t+o(\Delta t)$。

（4）每个错误发生的机会相同，且严重等级相同，各失效之间相互独立。

（5）软件的运行方式与预期的运行方式相似。

MO模型是失效强度函数随失效发生而指数递减的非齐次泊松过程。失效强度函数随失效发生指数递减说明早期发现的缺陷比晚期发现的缺陷对失效强度函数的减小作用大。之所以称之为对数泊松模型，是因为期望的失效数是时间的对数函数。

2. 模型形式

由模型假设可知软件失效均值函数$m(t)$和失效强度函数$\lambda(t)$分别为

$$m(t)=\frac{1}{\theta}\ln(\lambda_0\theta t+1) \tag{5.20}$$

$$\lambda(t)=m'(t)=\frac{\lambda_0}{\lambda_0\theta t+1} \tag{5.21}$$

定义t_i为第i次失效发生的时间，x为从第i次失效开始前的执行时间，则软件失效$i-1$次后的可靠度函数为

$$R(x|t_{i-1})=\left[\frac{\lambda_0\theta t_{i-1}+1}{\lambda_0\theta(x+t_{i-1})+1}\right]^{\frac{1}{\theta}} \tag{5.22}$$

t_{i-1}时刻后的软件平均失效等待时间为

$$\mathrm{MTTF}_i=\int_0^{\infty}R(x|t_{i-1})\mathrm{d}x=\int_0^{\infty}\left[\frac{\lambda_0\theta t_{i-1}+1}{\lambda_0\theta(x+t_{i-1})+1}\right]^{\frac{1}{\theta}}\mathrm{d}x \tag{5.23}$$

3. 参数估计

MO模型需要利用完全数据进行参数估计，通过最大似然估计推导出下列方程：

$$\frac{n}{\beta_1}-\sum_{i=1}^{n}\frac{t_i}{\beta_1 t_i+1}-\frac{nt_n}{(\beta_1 t_n+1)\ln(\beta_1 t_n+1)}=0 \tag{5.24}$$

其中$\beta_1=\lambda_0\theta$。

通过上述方程得到β_1的最大似然估计$\hat{\beta}_1$，然后利用公式$m(t_n)=n$得到参数θ和λ_0的估计值：

$$\begin{cases}\hat{\theta}=\dfrac{1}{n}\ln(\hat{\beta}_1 t_n+1)\\ \hat{\lambda}_0=\hat{\beta}_1/\hat{\theta}\end{cases} \tag{5.25}$$

5.1.4 Inflection S形模型

1984年，Ohba提出了Inflection S形模型。在该模型中，软件可靠性增长曲线呈S形，即开始增长比较慢，然后迅速增长，直到达到极限。

1. 模型假设

Inflection S形模型有以下假设：

（1）程序在同实际执行环境相差不大的条件下执行。

（2）部分故障在其他故障被发现之前不会被发现。

（3）任何时间内故障的发现率与当前软件中残留故障数成正比。

（4）被隔离的故障可以被完全排除。

（5）故障检测率是一个不变的常数。

2. 模型形式

由假设易知，

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b(t)(a(t)-m(t)) \tag{5.26}$$

其中，$b(t)$为故障检测率函数，$a(t)$为软件的总故障函数。

进一步假设故障检测率函数为$b(t)=b/(1+\beta\mathrm{e}^{-bt})$，其中，$b$与$\beta$分别表示故障检测率

因子与变点因子(inflection factor)。由此可得

$$m(t)=\frac{a(1-\mathrm{e}^{bt})}{1+\beta\mathrm{e}^{-bt}} \tag{5.27}$$

其中，a表示初始故障数。该模型的失效强度函数为

$$\lambda(t)=\frac{ab(1+\beta)\mathrm{e}^{-bt}}{(1+\beta\mathrm{e}^{-bt})^2} \tag{5.28}$$

3. 参数估计

Inflection S形模型参数估计的方法和GO模型相同。

1）基于故障间隔时间进行参数估计

利用故障间隔时间进行参数估计的方程式如下：

$$\begin{cases}\hat{a}=\dfrac{N(1+\mathrm{e}^{-\hat{b}S_N})}{1-\mathrm{e}^{-\hat{b}S_N}}\\ \dfrac{NS_N\mathrm{e}^{-\hat{b}S_N}(1+\hat{\beta})}{(1-\mathrm{e}^{-\hat{b}S_N})(1+\hat{\beta}\mathrm{e}^{-\hat{b}S_N})}=\dfrac{N}{\hat{\beta}}-\displaystyle\sum_{i=1}^{N}S_i+2\sum_{i=1}^{N}\frac{\hat{\beta}S_i}{1+\hat{\beta}\mathrm{e}^{-\hat{b}S_i}}\end{cases} \tag{5.29}$$

求解式(5.29)即得出各个参数的估计值。

2）基于累积故障数进行参数估计

利用累积故障数进行参数估计的方程式如下：

$$\begin{cases}\hat{a}=\dfrac{y_N(1+\beta\mathrm{e}^{-\hat{b}t_N})}{1-\mathrm{e}^{-\hat{b}t_N}}\\ \dfrac{y_Nt_N\mathrm{e}^{-\hat{b}t_N(1-\hat{\beta}+2\hat{\beta}\mathrm{e}^{-\hat{b}t_N})}}{(1-\mathrm{e}^{-\hat{b}t_N})(1+\hat{\beta}\mathrm{e}^{-\hat{b}t_N})}\\ =\displaystyle\sum_{k=1}^{N}(y_k-y_{k-1})\left(\frac{t_i\mathrm{e}^{-\hat{b}t_i}-t_{i-1}\mathrm{e}^{-\hat{b}t_{i-1}}}{\mathrm{e}^{-\hat{b}t_i}-\mathrm{e}^{-\hat{b}t_i}}+\frac{\hat{\beta}t_i\mathrm{e}^{-b\hat{b}t_i}}{1+\hat{\beta}\mathrm{e}^{-\hat{b}t_i}}+\frac{\hat{\beta}t_{i-1}\mathrm{e}^{-\hat{b}t_{i-1}}}{1+\hat{\beta}\mathrm{e}^{-\hat{b}t_{i-1}}}\right)\end{cases} \tag{5.30}$$

求解式(5.30)即得出各个参数的估计值。

5.2 NHHP类软件可靠性增长模型

在软件测试过程中，随着故障不断被检测出来并被排除，软件可靠性持续获得增长，这为软件可靠性研究提供了有效的切入点。软件可靠性增长模型（SRGM）从软件失效的角度进行可靠性建模，采用以微分方程(组)为主的数学手段建立软件测试过程中的若干随机参量间的定量函数模型，这些随机参量包括测试时间、累积检测的失效或修复故障个数、测试工作量（Testing-Rffort, TE）等。基于求解获得的累积检测故障数量函数$m(t)$可以获得测试阶段的软件可靠性。因此，建立能够准确描述真实随机测试过程的累积检测故障数量函数$m(t)$成为SRGM研究的关键。

5.2.1 软件可靠性增长模型建模过程

面对自故障检测至排除的软件测试过程，基于不同的假设可以建立形式各异的数学模型，产生不同的结果。从软件可靠性建模流程角度看，不同SRGM的构建都可以用图5.2所

示的过程描述。

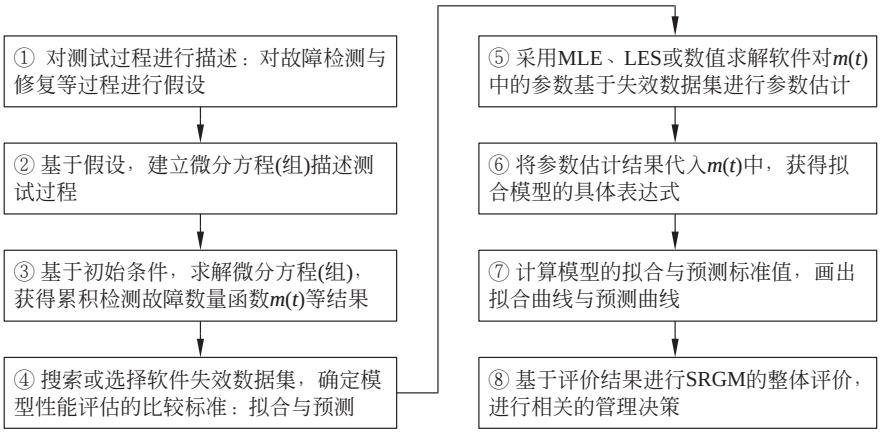

图 5.2 SRGM 建模过程

从图5.2可以看出，SRGM 建模的机理是对测试过程中的各随机变量建立适当的数学模型，以指导测试资源的使用，以便在测试周期内提高软件可靠性。数学模型的有效性主要通过在真实的失效数据集上进行拟合与预测加以验证。由于对测试过程的认知存在差异性，因此，众多 SRGM 模型被提出。

SRGM 主要通过指数型（exponential）或者 S 形模式描述软件失效现象，且本质上二者均是基于非齐次泊松过程（NHPP）的一种随机过程，由此衍生出多种 SRGM。在这些模型中，NHPP 类 SRGM 最具有吸引力，应用也最为广泛。现有的上百个 NHPP 类 SRGM 均基于以下假设：

（1）失效事件随机发生，测试人员与排错人员对失效的观察与故障的排除过程服从 NHPP。

（2）设 $\{N(t), t \geqslant 0\}$ 为随机计数过程，$N(t)$ 为 $[0,t]$ 内测试人员累积检测的故障数量，且 $E[N(t)] = m(t)$ 为均值函数，满足 $m(0) = 0$。

利用 NHPP 的基本性质，可以得到

$$P[N(t) = k] = \frac{m(t) \times \mathrm{e}^{-m(t)}}{k!}, \quad k = 0, 1, 2, \cdots \tag{5.31}$$

$$m(t) = \int_0^t \lambda(\tau)\mathrm{d}\tau \tag{5.32}$$

测试阶段的软件可靠性可表示为

$$\begin{aligned} R(x|t) &= P[N(t+x) - N(t) = 0] \\ &= \mathrm{e}^{-(m(t+x)-m(t))} \end{aligned} \tag{5.33}$$

若设 $t(t \geqslant 0)$ 为上次失效发生的时刻，对于给定的时长 $x(x > 0)$，则软件可靠性在 $(t, t+x]$ 内可表示为式 (5.33)。

这里，以经典的 GO 模型为例，其基本假设如下：

（1）软件失效过程服从 NHPP。

（2）t 时刻，累积检测到的故障数量 $m(t)$ 的变化率与当前剩余的故障数量 $a - m(t)$ 成正

比，比例系数为b。

由上述基本假设可得到下面的微分方程：

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}=b(a-m(t)) \tag{5.34}$$

其中，a为软件中的总故障个数。在$m(0)=0$的初始条件下，可求得$m(t)=a(1-\mathrm{e}^{-bt})$。

SRGM的关键是在某些假设条件下建立微分方程，求解得到$m(t)$，而不同SRGM的差异直接体现在$m(t)$上。由式(5.33)可以看出，$R(x|t)$是$m(t)$的函数，因而软件可靠性随时间的增长完全取决于$m(t)$。而$m(t)$与真实失效个数的拟合与预测情况（即接近情况）是衡量SRGM性能优劣的根本标准。这样，获得特定测试环境下合适的$m(t)$是SRGM研究的核心内容。

5.2.2 影响SRGM的关键参数因素分析

真实测试环境的不稳定直接导致了SRGM需要把实际的随机因素考虑进去，从关注失效个数的发现直至故障被排除的角度提高软件可靠性，是SRGM用于建模描述软件测试过程最主要的线索。SRGM旨在探索软件测试过程中可靠性同与其直接相关的决定因素间的内在机理，即故障检测至排除过程对软件可靠性的定量影响关系，为提高可靠性、确定成本结构和软件最优发布等提供决策依据。

由2.3节可知，软件中总故障个数$a(t)$和故障检测率$b(t)$是影响SRGM的重要参数因素。另外，测试工作量（Testing Effort, TE）用测试工作量函数（Testing-Effort Function, TEF）表征，即TEF同样是影响SRGM的重要参数因素。

1. 建模描述排错目标——总故障个数

软件测试与排错人员希望将全部故障排除，获得高可靠性，但这并非易事，也不现实，主要原因如下：

（1）软件中总故障个数未知（虽可以假定为有限或无限个数），根本无法确定。

（2）在排错过程中因排错不彻底以及引入新故障等使得总故障个数会增加。

因而，设定$a(t)=a$，忽略了排错中人为引入错误的事实。为此，设定$a(t)$为t的某种增函数形式较为常见，将其归为表5.3 中的3大类型。

表 5.3 软件中总故障个数$a(t)$类型

类 型	软件中总故障个数$a(t)$	概要分析
乐观类型 (有限故障)	$a(t)=c+a(1-\mathrm{e}^{-t})$	总故障个数有限，当$t\to+\infty$时$\lim\limits_{t\to+\infty}a(t)=c+a$
悲观类型 (无限故障)	$a(t)=a\mathrm{e}^{\alpha t}$ $a(t)=a(1+\alpha t)$ $a(t)=a(1+\beta t)$	$a(t)$随测试时间t持续增长至无穷大
中间类型	$a(t)=a+\beta m(t)$ $a(t)=a\mathrm{e}^{\alpha}W^{*}(t)$	由于$m(t)$与测试开销$W^{*}(t)$通常为增函数，但存在有限增长的可能，故$a(t)$增长也可能有限度

这里，有两点需要特别指出：

（1）悲观类型虽不易被接受，且根本无法在实际中验证，但其在拟合真实失效数据时较乐观类型表现好。这或许可解释为：对于实际软件，无法进行无限次的测试，因此软件

中的故障个数无法准确估计。

（2）这些$a(t)$考虑到了故障个数增加的情况，但直接导致其增加的原因正是排除故障过程，因而，$a(t)$的变化率应与累积修复的故障数量变化率成正比。

2. 测试环境描述能力——故障检测率FDR

故障检测率（Fault Detection Rate, FDR）用于描述当前测试环境下故障被检测出的比率，其与整体测试策略（技术、工具、测试人员技能等）紧密相关。由于测试策略的多样性，使得FDR存在多种函数，且均为研究人员自行设定。

早期研究中认为FDR为常量，设定$b(t)=b$。随着研究的深入，FDR已呈现出多种函数形式，例如：

$$b(t)=b^2t/(1+bt)$$
$$b(t)=\frac{b}{1+\beta \mathrm{e}^{-bt}}$$
$$b(t)=bt^d+\delta\gamma(t)$$
$$b(t)=b\alpha\beta \mathrm{e}^{-\beta t}(t)$$
$$b(t)=b\alpha\beta t\mathrm{e}^{-\beta t^2/2}$$

鉴于FDR是测试环境的直接描述，因而测试环境的改变可借助FDR呈现。因而，当前在考虑变动点（Change-Point, CP）的SRGM研究中，多从建立FDR分段函数的形式实施。

3. 成本支持下的SRGM研究——考虑测试工作量

为实现预期发布，测试过程需要消耗测试资源作为保障。测试工作量函数TEF是对测试资源（如测试时间、测试用例数等）消耗的数学描述，表5.4列出了主要的测试工作量函数。

表 5.4 主要的测试工作量函数

模　型	公式化描述	说　明
Logistic TEF	$W_k(t)=\dfrac{W}{1+A\mathrm{e}^{(-\alpha kt)}}$	最早被提出的Logistic类型的TEF
Generalized logistic TEF	$W_k(t)=\dfrac{W}{\sqrt[k]{1+A\mathrm{e}^{(-\alpha kt)}}}$	依据参数k设置的不同，$W_k(t)$存在多种形式。若$k=1$，则演变为Logistic TEF形式
	$W_{k,\beta}(t)=\dfrac{\sqrt[k]{\frac{k+1}{\beta}}}{\sqrt[k]{1+A\mathrm{e}^{(-\alpha kt)}}}W$	若$\beta=k+1$，则$W_{k,\beta}(t)$演变为上面的$W_k(t)$
Deformable logistic TEF	$W(t)=\dfrac{\mathrm{e}^{\alpha t}-v}{\mathrm{e}^{\alpha t}+u}W$	初值不为0的改进Logistic TEF
Weibull TEF	$W_k(t)=(1-\mathrm{e}^{-\beta t^{\delta}})^{\theta}W$ $W>0,\beta>0,\delta>0,\theta>0$	根据其中参数的不同设置，Weibull TEF又可具体分为5种特定的类型
Log-Logistic TEF	$W_{\mathrm{ll}}(t)=\dfrac{(\theta t)^{\delta}}{1-(\theta t)^{\delta}}W$	初值不为0的Log型Logistic TEF

5.2.3 统一的SRGM框架模型

不同的SRGM差异较大，难以辨识，统一建模方法的提出旨在屏蔽这些差异，采用框架技术是近年来软件可靠性研究的一个主题。按照对实际测试过程中的认识，当考虑到多

种可能的测试环境时，这些SRGM在建模中主要涉及不完美排错、测试工作量、变动点等。事实上，不同SRGM的本质区别在于对测试阶段认识的差异上，具体表现为建立的微分方程（组）的不同。因此，一种趋势是建立相对统一的SRGM框架模型，使其能够涵盖多种现有的模型：

$$\begin{cases} \dfrac{\mathrm{d}m(t)}{\mathrm{d}t} = \dfrac{\mathrm{d}W(t)}{\mathrm{d}t}\left[b(t)\left(a(t)-c(t)\right)\right] \\ \dfrac{\mathrm{d}a(t)}{\mathrm{d}t} = r(t)\dfrac{\mathrm{d}c(t)}{\mathrm{d}t} \\ \dfrac{\mathrm{d}c(t)}{\mathrm{d}t} = p(t)\dfrac{\mathrm{d}m(t)}{\mathrm{d}t} \end{cases} \tag{5.35}$$

该统一的SRGM框架模型的建立基于下面的假设条件：

（1）累积检测的故障数量$m(t)$变化率与当前测试工作量消耗率及故障检测率$b(t)$下剩余的故障数量成正比。

（2）故障引入率与修复过程中的修复率成比例，比例系数为$r(t)$。

（3）修复率与检测过程中的检测率成比例，比例系数为$p(t)$。

显然，由于对测试过程的认识差异直接决定了建立的微分方程（组）的不同，这样求得的$m(t)$等变量就会存在差异。式(5.35)可以涵盖现有多个SRGM，据此给出SRGM的一种分类。

在式(5.35)中，当不考虑第2子式与第3子式，忽略测试工作量，认为检测率$\mathrm{d}m(t)/\mathrm{d}t$与尚未检测到的故障数量成正比，且设定$b(t)$与$a(t)$均为常量，即$a(t)=a, b(t)=b$时，可以得到经典的GO模型：

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b\left(a-m(t)\right) \tag{5.36}$$

易见，GO模型忽略了测试中较多的实际因素，然而它是最早的SRGM，同时期的其他模型在建模上也较为粗糙。

1. 考虑实际因素的SRGM

若式(5.35)中不考虑第2子式与第3子式，忽略测试工作量，且认为检测率$\mathrm{d}m(t)/\mathrm{d}t$与尚未被检测到的故障数量$(a(t)-m(t))$成比例，可以得到典型的不完美排错（Imperfect Debugging, ID）模型——Pham模型，其建立的微分方程为

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t} = b(t)[a(t)-m(t)] \tag{5.37}$$

该模型的提出者Pham认为，$a(t)=a(1+rt)$，即$a(t)$是测试时间t的线性增函数。这表明测试中会引入新故障，因而称该模型为不完美排错模型；$b(t)=\dfrac{b(1+\sigma)}{1+\sigma\mathrm{e}^{-b(1+\sigma)t}}$，其中，$\sigma$是反映测试人员技能的学习因子。

而当$a(t)=a, b(t)=b$时，式(5.37)演变为最早的GO模型。由于$a(t)$与$b(t)$可灵活设置，这样，式(5.37)是框架模型。例如，令$b(t)=b\left(\gamma+(1-\gamma)\dfrac{m(t)}{a}\right)$，其中，$\gamma$为弯曲因子，表示软件中不相关故障所占的比例。当$\gamma=1$时得到的$m(t)$即为指数型SRGM，其余情况得到的是S形模型。这种转化现象被称为柔韧性，通常框架模型都具有这种柔韧性。

2. 融合测试工作量的SRGM

故障检测与修复是在测试工作量的消耗下进行的，测试工作量可有效地描述软件开发过程中的工作剖面，因而SRGM中应考虑到测试工作量的存在。

这样，在式(5.37)的基础上，当考虑测试工作量时，可以得到下面被众多文献所采用的模型：

$$\frac{\mathrm{d}m(t)}{\mathrm{d}t}\frac{1}{w(t)}=b(t)\left[a-m\left(t\right)\right] \tag{5.38}$$

由于式(5.38)中考虑到了测试工作量，因而它能够将测试资源的影响纳入SRGM中。易知，由于$W(t)=\int_0^t w(t)\mathrm{d}t$存在多种形式，式(5.38)也是一种框架模型。

3. 考虑不完美排错与测试工作量的SRGM

由于简化建模的需要，很多SRGM做了一些偏离实际的假设。若研究中的假设条件更加靠近真实的测试过程，则建立的SRGM可被称为不完美排错模型。若式(5.35)中不考虑第3个子式，且认为新故障引入率与检测率成正比，则得到Huang模型：

$$\begin{cases}\dfrac{\mathrm{d}m\left(t\right)}{\mathrm{d}t}\dfrac{1}{w\left(t\right)}=b\left(t\right)\left[a\left(t\right)-m\left(t\right)\right]\\ \dfrac{\mathrm{d}a\left(t\right)}{\mathrm{d}t}=r\dfrac{\mathrm{d}m\left(t\right)}{\mathrm{d}t}\end{cases} \tag{5.39}$$

这里，认为故障修复概率为100%，新故障会在检测过程中被引入。更进一步，在式(5.39)的基础上，设定故障检测率函数：$b\left(t\right)=b\left[h+\left(1-h\right)\dfrac{m\left(t\right)}{a\left(t\right)}\right]$，可得到Ahmad模型。

显然，建立的不完美排错模型越靠近真实的测试环境，则模型就越准确，但同时求解方法已开始过渡到复杂的非解析方法。

4. 考虑不完美排错、测试工作量和变动点的SRGM

与前面相比，这是一种较为全面的建模，不仅涉及不完美排错、测试工作量，还将测试环境的改变而引发的变动点融入SRGM中。受到多种因素的影响（运行环境、测试策略、失效密度以及资源分配等），软件失效分布并不均匀，这就是引入变动点的客观因素。这样，在式(5.35)中，若不考虑第2子式与第3子式，设定$a(t)=a$，且从$b(t)$的角度考虑变动点，则得到GLTECPM模型：

$$\begin{cases}\dfrac{\mathrm{d}m\left(t\right)}{\mathrm{d}t}\dfrac{1}{w\left(t\right)}=b\left(t\right)\left[a-m\left(t\right)\right]\\ b\left(t\right)=\begin{cases}b_1, & 0\leqslant t\leqslant\tau\\ b_2, & t>\tau\end{cases}\end{cases} \tag{5.40}$$

其中，τ是变动点。

相比之下，从$W(t)$的角度可以考虑多个变动点，得到下面的模型：

$$\begin{cases}\dfrac{\mathrm{d}m\left(t\right)}{\mathrm{d}t}\dfrac{1}{w\left(t\right)}=b\left(t\right)\left[a-m\left(t\right)\right]\\ W\left(t\right)=\begin{cases}W\left(1-\mathrm{e}^{-\beta_1 t}\right), & 0\leqslant t\leqslant\tau_1\\ W\left(1-\mathrm{e}^{-\beta_1\tau_1}+\mathrm{e}^{-\beta_2\tau_1}-\mathrm{e}^{-\beta_2 t}\right), & \tau_1<t\leqslant\tau_2\\ \cdots\end{cases}\end{cases} \tag{5.41}$$

其中，被分段讨论的$W(t)$为Yamada指数型TEF：$W(t)=W(1-\mathrm{e}^{-\beta t})$，当然也可以是其他类型的TEF。

相比于测试用例作为故障移除周期单位的离散时间模型，上面的以时间t作为自变量的SRGM被统称为连续时间模型，且后者居多。

这些模型的提出，极大地丰富了SRGM的研究，在缺少安全性和可信性增长模型的情况下，使得SRGM成为唯一能够描述可靠性随时间提高的技术手段。这里有3点需要指出。

（1）当前，对不完美排错的认识主要停留在排错的不彻底和新故障的引入上。实际上，上述两种情况只是真实测试环境中很狭义的不完美，此外更多的是广义不完美。因而，不完美排错的概念与研究范畴理应进行更大的扩展。

（2）虽然这些模型最终求解得到的$m(t)$形式各异，但均是基于对测试过程进行不同假设后建立方程求解的结果。从这个角度看，这种解析方法存在着一个固有的不足：数学方程只能在有限范围内进行建模，因为复杂模型既难以提出又难以求解，因而新的技术手段，例如仿真，就在此背景下应运而生。

（3）上述模型对测试过程的认识虽各有不同，但均只是局限在单纯的采用微分方程（组）建模并求解的范畴内。

5.3 习题

1. 简述软件可靠性建模的具体步骤。

2. 使用基于执行时间的模型（Musa）和MO模型对软件可靠性进行评估时，需要选取的参数是什么？

3. 对于如下微分方程：

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

证明均值函数为

$$m(t)=\mathrm{e}^{-B(t)}\left[m_0+\int_{t_0}^{t}a(\tau)b(\tau)\mathrm{e}^{B(\tau)}\mathrm{d}\tau\right] \tag{5.42}$$

初始条件为$m(t_0)=m_0$，其中$B(t)=\int_{t_0}^{t}b(\tau)\mathrm{d}\tau$，$t_0$为开始调试排错的时间。

4. 假设错误总数函数和错误检测率函数分别为

$$a(t)=\alpha_2(1+\gamma t),\quad b(t)=\frac{b^2t}{bt+1}$$

试根据式(5.42)证明均值函数具有如下形式：

$$m(t)=\alpha_2(1+\gamma t)-\frac{bt+1}{\mathrm{e}^{bt}}-\frac{(1+bt)\alpha_2\gamma}{b\mathrm{e}^{bt+1}}\left[\ln(bt+1)+1+\sum_{i=1}^{\infty}\frac{(bt+1)^{i+1}-1}{(i+1)!(i+1)}\right]$$

5. 假设故障总数函数和故障检测率函数分别为

$$a(t)=\alpha(1+\gamma t)^2,\quad b(t)=\frac{\gamma^2t}{\gamma t+1}$$

根据式(5.42)证明均值函数具有如下形式：

$$m(t)=\alpha\left(\frac{1+\gamma t}{\gamma t}\right)(\gamma t\mathrm{e}^{\gamma t}+1-\mathrm{e}^{\gamma t})$$

6. 某实时指控系统的故障数据集如表5.5所示。

（1）根据所给数据，计算GO模型的参数a和b的最大似然估计。

（2）求均值函数$m(t)$和可靠度函数。

（3）在时间间隔$[10,12]$内不会发生失效的概率是多少？

（4）选择另一个NHPP软件可靠性模型，并重做（1）～（3）。该模型与GO模型哪个好？解释原因并证明你的结论。

表 5.5 某实时指控系统的故障数据集

失效时间	失效数	累计失效数	失效时间	失效数	累计失效数
1	27	27	6	7	82
2	16	42	7	2	84
3	11	54	8	5	89
4	10	64	9	4	92
5	11	75	10	1	93

7. 假设随时间变化的故障总数函数和故障检测率函数分别为

$$a(t)=c+a(1+\mathrm{e}^{-\alpha t}),\quad b(t)=\frac{b}{1+\beta\mathrm{e}^{-bt}}$$

（1）根据式(5.42)给出其均值函数和可靠度函数。

（2）使用表5.1给出的NTDS数据集，并假设$t_0=149, m(149)=22$，利用最大似然估计对参数a、b、c、α和β进行估计。

（3）利用最大似然估计的结果预测下一个失效的时间t_{23}。

8. 利用表5.1的NTDS数据集中的前25个失效数据。

（1）计算两个与不完美排错相关的软件可靠性增长模型的未知参数的最大似然估计。

（2）求出这两个软件可靠性增长模型的均值函数和可靠度函数。

（3）比较这两个软件可靠性增长模型的好坏，解释原因并证明你的结论。