Overview: Software Reliability Growth Models

概论：软件可靠性增长模型

摘要

软件可靠性是软件质量的一个重要属性。可靠性是程序执行其应有功能的能力，而可用性是一个系统可运行和可访问的程度。在过去 30 年中已经建立了许多数学模型用来估计软件产品的可靠性增长，这些模型通常称为 SRGM。本论文主要描述这些 SRGM 的概要和适用性。

术语

based on the failure history 基于失效历史
error seeding model 错误根植模型
fault counting models 故障计数模型
fault detection rate 故障探测比率
fault removal technique 故障移除技术
input domain model 软入域模型
operational profile
Poisson distribution 泊松分布
reliability factor 可靠性因子 RF
residual errors 残留错误
software reliability growth models 软件可靠性增长模型（SRGM）
time between failure modes 失效间隔模型
undetected errors 未探测到的错误
not seeded errors detected 探测到的未根植错误
seeded errors detected 探测到的根植错误

软件可靠性增长模型

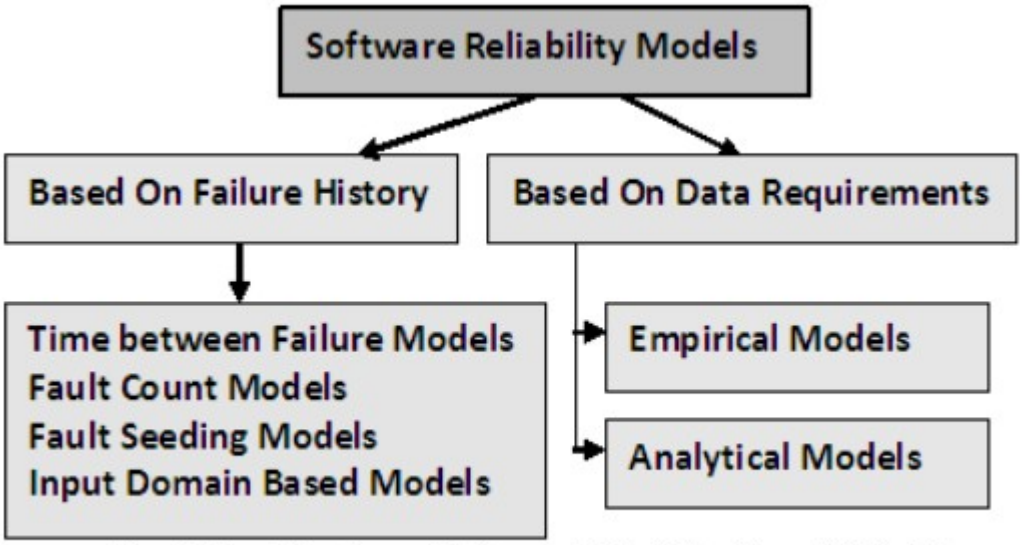

Fig. 1 Classification of Software Reliability Growth Models

SRGM 主要分为两类：一是基于失效历史；二是基于图中的数据需求。本论文主要是描述前者，其模型的主要因子全部列在图 2 中。

Model Category	Data Requirements
Time Between Failure	Time at each failure and failures are recorded
	Operational Profile and corresponding tests
Failure Count	Failure count in the interval and length of interval
Fault Seeding	Number of seeded faults
Category	Types and number of fault discovered

Fig. 2 Data requirements of the SRGM Models

1 失效间隔模型

这些模型主要是基于失效之间的间隔，它们假设相邻两次失效之间的时间是一个随机数，其遵循某种分布，该分布的参数取决于程序中残留故障的数目。这些参数的估算是从失效之间的间隔、到下一次失效的平均时长这些观察到的数据获得的。

1.1 Jelinski Moranda Model

JV 模型是一个指数模型。该模型中，测试一开始有 N 个软件故障，每一个都相互独立，并且等概率地可引起测试故障。可以使用故障移除技术来清除缺陷，使得调试过程中不再引入新的缺陷。该模型的基本假设是：

- 代码行的缺陷数目是常量的
- 软件的 operational profile 是一致的
- 在软件运行过程中碰到每一个故障的概率是等同的
- 故障发生之间的故障探测比率仍然是一个常量
- 故障探测比率与软件的当前故障内容是成比例的
- 每一个探测到的错误可以立即纠正
- 失效是彼此独立的

模型的公式如下：

$$\mu(t_j) = 1/b\,(a-(j-1))$$

$\mu(t_j)$表示预测的错误数目，其中 b 表示波形因子（在哪个斜率上失效率下降），a 是软件错误的总体数目，t_j表示第 j 个故障的发生时间。那么，如果所有错误是可以探测出来的话，残留错误可以计算出来：

$$ER = a - \mu(t_j)$$

可靠性因子可用来度量软件可靠性，它的有性值是 0 ~ 1 之间。如果 RF = 1，那么考虑中的软件是完美的；如果 RF = 0，那么软件就是高度易受损的。如果 RF 接近 1 时，那么软件就认为是可靠的了。其中可靠性因子计算如下：

$$RF = 1 - (ER/a)$$

2 失效计数模型

该模型主要基于指定时间间隔内发生的失效（故障）次数。该模型假设失效次数遵循一个已经的随机过程。以时间为横轴的泊松分布会是离散的或者连续的失效率。横轴的时间可能是月份也可是 CPU 时间。失效率参数可以从观察到的失效次数估算到，那么软件可靠性参数就可以从适当的公式计算获得。

2.1 Goel- Okumuto Non- homogeneous Poisson Process Model

该模型中，Goel-Okumuto 假设一个软件系统受制于系统故障所引起的失效，其出现次数是随机的。而 Non- homogeneous Poisson Process 模型将单位时间内的故障数目视为独立的泊松随机变量。该模型的基本假设如下：

- 故障的累计数目随着时间 t 的变化遵循泊松过程
- 对于时间间隔的任一个有限集合，单个时间间隔内探测到的故障数目是独立的
- 缺陷一旦发现就会立即修复
- 修复缺陷是完美的，也就是说本次的缺陷修复不会引入新的缺陷
- 在测试过程中不再增添新的代码
- 相同的时间内执行相同的代码，单位时间内发现一个缺陷的概率是相等的

模型的公式如下：

$$\mu(t) = a(1 - e^{-bt})$$

$\mu(t)$表示累计的失效次数，a 是期望的失效总次数，而 b 是每个故障的故障探测率，是一个常量且 b>0。

3 错误或失效根植模型

在这种模型中，人为产生的错误都是程序代码的各种组合定义好的。之后，测试运行就是用来探测错误的，基于已探测到的错误总数来检查实际错误数目与人为错误之间的比例。自然地，人为错误有多少，对于测试来讲是未知的。未探测到的错误数目如下计算：

$$FU = FG \cdot (FE / FEG)$$

FU 表示未探测到的错误数目，FG 表示探测到的未根植错误，FE 表示根植的错误数目，FEG 表示探测到的根植错误数目。可以通过测试来知道实际上存在多少错误，这个模型假设如下：

- 已根植的错误在程序中是随机分布的
- 已根植的错误是等概率地被探测到的

Mills Hyper 几何模型

该模型假设已知故障的数目是随机地根植在程序中的。原生的故障总数目可以通过测试过程中发现的故障数目、同时使用 Mills Hyper 几何分布来估算。

4 基于输入域的分类

该模型通过发现程序中的所有唯一路径，然后执行每一条路径，从而保证所有的都是可以被测试的。其代码是 Nelson 模型，该模型假设如下：

- 输入 profile 分布是已知的
- 使用随机测试
- 输入域可以分解为等价类

Nelson 模型

In this model, reliability of software is calculated by executing the software for a sample of n input. Inputs are randomly selected for the input domain set S= (Si,i=1,.....,N) and each Si is set of data values required for the execution. Probability distribution Pi I ; the set (Pi, i = 1, N) is the operational profile or simply the user input distribution. And random sampling is done according to this probability distribution. Suppose n e is the number of execution that leads the execution to fail. Then estimation of reliability R1 is: $R1=\{1-n_e /n\}$.

结论

本篇论文首先介绍了已有的软件可靠性模型及其假设。然后，通过一个例子来分析这些模型的适用性。通过以上评估可以看出，随着残留错误的减少，可靠性因子在增加； 可靠性因子越接近 1，可靠性越高。与故障计数模型相比，失效间隔模型的假设前提更难以满足。