软件开发 可靠性 模型 ⑦

计算机科学1993Vol.20№.6

TP311.52

42-47

软件可靠性模型综述

童远琳（中国地质大学计算机科学与技术系 武汉430074）

摘 要

In this paper, the review for history of software reliability and models of software reliability is presented. Especially, the general theory of software reliability model and it's new advances in recent years are discussed. Furthermore, some basic criteria of model comparison are introduced.

一、引言

随着软件开发与研究的发展，软件产品的质量已成为一个不可忽视的问题，并由此推动了软件可靠性研究与应用。软件可靠性不仅是软件质量中最重要的定量描述指标，而且是面向用户的视图[1]。从软件可靠性概念出发，人们已先后建立起若干软件可靠性模型，并提供了一些评估软件可靠性的有效方法。

软件可靠性的研究可追溯到六十年代。1967年Hudson首先提出将软件开发过程视为一个生命周期，它以软件开发过程中产生差错而开始，以差错修正而结束[1]。残留的故障数定义了过程状态，转换概率亦特征地描述了生命周期起始和结束的功能。Hudson的分析之局限在于：当发现的差错率同残留的差错数及时间的正幂次方成比例时，他的纯数学假定将造成结束的处理。Hudson的研究表明，发现的差错数服从于二项分布，其均值是时间的Weibull函数的形式。

继Hudson之后，Jelinski和Moranda在1972年做了进一步的研究。同年，Shooman也做了类似的工作。他们的研究都假定有一个故障的分段常数，危险率是同残留差错数成比例的。危险率在每一次由一常数量进行的差错修正中发生变化，但在两次修正期间不变。前者采用最大似然估计来确定软件中的差错总量及残留差错数和危险率之间的比例常数。后者假定危险率同以下几个参量成比例：每条指令的差错密度；在每一个单元时间内执行单指令的数目及大容量常数(bulk constant)。

1972年Schneidewind提出一种软件可靠性模型。他主张依据不同的可靠性函数来选择最适合项目中问题的分布函数。这些分布函数有指数分布函数、正态分布函数、γ分布函数及Weibull分布函数。以后，他又提出将每个时间间隔内发现差错视为一种指数均值函数的非齐次泊松过程，并采用了最小二乘法或最大似然估计法来确定过程中的参数。

Schick和Wolverton于1973年提出了一种软件可靠性模型，认为危险率同残留差错数和时间的乘积成比例。因此，随时间的增长，危险率在差错修正中变化的范围加大。五年以后，他们又给出了修正模型，其危险率是时间的抛物线性函数，而不是时间的线性函数。这种危险率近似于Weibull分布，但并不等同。

1973年Littlewood和Verall采用贝叶斯方法进行软件可靠性测试。他们将危险率视为随机变量，其分布函数含有一个随经历的

故障数变化的参数。Keiller等在1983年也提出了类似的模型，只是使用了不同的参数来表达可靠性变化。

Musa 1975年在总结前人工作的基础上，提出了可靠性模型的执行时间模型。他假定执行时间是在程序中设置的最实际的故障诱发重点测试，而计时时间不能说明程序在运行或测试中的不同用途。随后，1984年Musa和Okumoto建立了基于非齐次的泊松过程的对数泊松模型，该过程有随期望经历的故障数递减的密度函数，反映了在早期执行阶段中差错修复的状况。

可靠性模型的早期研究主要是观察不同性能的模型。七十年代末至八十年代初，研究开始集中于对软件可靠性模型进行比较和客观的最好选择。早期工作的不足之处是缺少较好的故障数据和所需的比较准则。这一期间的进展主要应归于：

- Musa 1979年发表的合理的良好性能的数据集，促进了比较工作；
- Musa和Okumoto 1983年对软件可靠性模型的基本概念进行了检查并发展了分类条目。此项工作有助于分类和组织比较并导致新模型的发展；
- Iannino等1984年给出了令人满意的比较准则。

下面将着重讨论近年来发展起来的几种新的软件可靠性模型。

二、基本术语和概念

我们首先介绍本文所用的基本术语和概念。

1. 软件故障和软件差错

软件故障(software failure) 是指程序运行的输出结果和用户需求之间的偏差。故障发生在程序的执行期间内。潜在故障可由程序员通过设计检查、代码阅读或其它一些方法来发现。

影响故障行为的两个主要因素是：(1) 软件执行时的差错数。(2) 运行轮廓。

软件差错 (software fault) 指在一定条件下运行程序时，能够引发或潜在地引发故障的程序缺陷。一个差错很可能引发许多复杂的故障。开发者在进行最初的设计或在程序中加入新的细节以及在修正活动中都有可能引入新的差错。

排除差错必须依据执行时发生的故障。故障发生既依赖于执行时间，也依赖于运行轮廓。差错也可以不经运行而通过检查、汇编诊断、重查设计或代码以及阅读代码等手段发现。

软件中的差错数指设计引入的差错数和排除的差错数之差，即软件中残留的差错数。

四个能反映故障发生特性的量是：a)故障时间；b) 故障间隔时间；c) 在给定时间内累计故障数；d) 一个时间区间内的故障数。这四个量均是随机变量，分别与故障发生的概率有关，但有其变化范围。只有故障发生的次数及每次发生的概率是已知的。

2. 运行轮廓

在定义运行轮廓 (operational profile) 之前，先介绍几个相关的概念。

程序的执行通常出现相互等同的重复运行，从而构成一个运行类型。运行类型是由程序的输入状态或与该程序有关的输入变量值的集合来确定的。输入状态指存在于程序外部并被程序使用的变量。所有可能的输入状态的集合称为输入空间。类似地，输出变量指存在于程序之外并被程序定值的变量。

运行轮廓表示程序能够执行的运行类型及其各自发生概率的集合。我们可根据输入状态和运行类型发生的概率作出运行轮廓图。

3. 软件可靠性

软件可靠性是指在某给定时间内特定环境下程序无故障运行的概率。

特定环境包括对主机、操作系统、支撑软件、操作环境、成功定义，与主机每一接口细节、输入输出数据范围和速率以及完整细节和操作步骤的精确说明。

在定义中时间t是变量。有三种时间与软件可靠性相关，分别为：

1）执行时间。程序指令执行时需要占用处理机的时间；

2）计时时间。我们所熟悉的日常时间，如时、分、秒；

3）时钟时间。在一台运行的计算机上从程序开始运行到结束所耗费的时间，包括等待时间及程序执行时间。

我们可以用两种不同的观点看待时间与故障的关系。一是均值函数，二是故障密度函数。前者表示每个时间点所对应的故障累计值的平均值，后者是均值函数的变化率或单位时间的故障数。确切地说，故障密度函数是均值函数对时间的导数，是一个瞬时值。从直观上看，采用故障密度函数的方法来表达软件可靠性是合适的。

三、软件可靠性模型及其比较

软件可靠性模型通常作为一种随机过程模型被提出，受故障时间的概率分布或经历的故障数和随时间变化的随机过程影响。要描述不同故障过程的一般数学形式，可以通过评估（对故障数据采用统计推理方法）或预测（与软件产品特性相关的参数值和开发过程）的方法来决定所采用形式中的参数值的置信区间。下面我们先介绍几种近年来发展起来的模型，然后给出模型的比较准则。

1. 可靠性模型

Musa提出的执行时间模型（Execation-Time Modeling）是将故障密度的刻画和预测作为执行时间函数的模型。它采用基于马尔可夫过程的方法将模型分为两大类：泊松类模型和二项式模型。泊松类模型可以是有限故障类模型，也可是无限故障模型，这取决于它们是如何被指定的。二项式模型是有限故障类模型。

·泊松类模型。令M(τ)表示时间τ内经历的故障数，P表示概率，则M(τ)过程可由如下假设导出：

(1) $P[M(\tau)=0\,|\tau=0]=1$；

(2) 增量$M(\tau+\Delta\tau)-M(\tau)$与历史无关，即过程未来的$M(\tau+\Delta\tau)$的马尔可夫性能仅依赖于当前状态$M(\tau)$，独立于过去的$M(x)$，$x<\tau$。

(3) $[\tau,\ \tau+\Delta\tau]$中出现故障的概率为$\lambda(\tau)\Delta\tau+o(\Delta\tau)$，其中，$\lambda(\tau)$指过程的故障密度，$o(\Delta\tau)$满足条件$\lim\limits_{\Delta\tau\to 0}\frac{o(\Delta\tau)}{\Delta\tau}=0$，它说明当$\Delta\tau$充分小时，$\Delta\tau$的二阶或高阶影响可忽略不计；

(4) $[\tau,\ \tau+\Delta\tau]$中出现一个以上故障的概率是$o(\Delta\tau)$。

$M(\tau)$过程可由指定均值函数或故障密度函数来定义，其关系式为：

$$\mu(\tau)=\int_0^{\tau}\lambda(x)dx$$

这里$\mu(\tau)$是$M(\tau)$过程的均值函数。

对有限故障类的泊松模型，可直接由二项式模型导出[1]，但需修正二项式模型的假设，即程序在$\tau=0$时，残留差错总数是均值为w_0的泊松随机变量。

对于泊松类模型，其可靠性可由下述公式计算：

$$R(\tau_i'\,|\tau_{i-1})=exp\{-[\mu(\tau_{i-1}+\tau_i')-\mu(\tau_{i-1})]\}$$

其中，τ_i表示第i个故障发生的时间，τ_i'表示第i个故障预测发生的时间。

·二项式模型。二项式模型基于以下假定：

(1) 无论软件故障何时出现，引起故障的差错也同时被排除；

(2) 在程序中有u_0个差错；

(3) 每个由差错引起的故障在时间上是独立和随机出现的，每个故障的危险率$Z_a(\tau)$表示对所有差错的危险率是相同的。

对于二项式模型，其可靠性可由下式计算出：

$$R(\tau_i'\,|\tau_{i-1})=exp\left[-(u_0-i+1)\int_{\tau_{i-1}}^{\tau_{i-1}+\tau_i'}Z_a(\tau)d\tau\right]$$

二项式模型和泊松类模型之间最主要的差别在于前者没有一个固定的内在差错数，但内在差错期望值w_0是已知的；后者的参数w_0是随机变量，受诸多因素的影响。如果实际的修复过程出现在故障时间内，则两个模型均不适用，但对于不完全修复，泊松类模型更为适合。

计时时间模型（Calendar-Time Modeling）是对所经过的含有执行时间的计时时间进行刻画和预测可靠性的模型。它建立在"调试排错过程"模型的基础之上。调试排错模型重点考虑：

- 对给定的执行时间运行程序并标识与处理相应的故障量所使用的资源；
- 资源量的有效性；
- 资源在其处于极限期间能被使用的程度。

根据模型的使用点和经过的测试周期的剩余点，可通过将有效资源量假定为常数来简化模型过程。

对于弄清计时时间与执行时间的关系，首先要了解资源需求与执行时间的联系，这可用资源需求公式来表达：

$$x_r=\theta_r\tau+\mu_r u(\tau)$$

其中，r为资源（r=C，F，I）；x_r表示资源需求；τ是执行时间；θ_r表示关于执行时间的平均资源使用率；μ_r指每个故障的平均资源使用；$u(\tau)$为时间τ内故障的均值函数或期望值。资源使用是指一段时间内所用的资源，如一个人时。

由于均值函数实际上是执行时间的函数，所以资源需求x_r为执行时间函数，依赖于所用的软件可靠性模型的执行时间部分。

我们还经常碰到一种增加或附加的资源需求Δx_r的情况，它的资源需求可由下式表述：

$$\Delta x_r=\theta_r\Delta\tau+u_r\Delta u(\tau)$$

其中$\Delta\tau$表示测试时间增量，$\Delta u(\tau)$表示经历故障数的增量。

对软件工程测试步骤进行控制的资源有：故障校正人员、故障标识人员、计算机时间。

计时时间与执行时间的比$dt_r/d\tau$和一给定极限资源的资源使用与执行时间的比有如下关系：

$$dt_r/d\tau=(1/p_r\rho_r)\times(\partial x_r/\partial\tau)\quad\cdots(1)$$

其中，$dt_r/d\tau$表示计时时间与执行时间的瞬时率；ρ_r指资源r的利用；$p_r\rho_r$为资源r的有效量。

在执行时间的任一点都有一个处于资源极限的使用状态，这个资源是使式(1)的比值为最大的资源，通过的计时时间可由下列方程决定：

$$dt/d\tau=\max_r(dt_r/d\tau)$$
$$=\max_r\{(1/p_r\rho_r)\times[\theta_r+u_r\lambda(\tau)]\}\quad\cdots\cdots(2)$$

由执行时间与与计时时间的关系可知，计时时间模型从资源利用的角度来刻画对应的执行时间期内资源利用的程度，即考察是否达到极限状态，其可靠性的计算公式仍可归结到执行时间模型中所采用的公式。

Noushin Ashrafi和Oded Berman建立了两种新模型[2]，在考虑花费和可靠性条件下，为选择合适的软件提供优化模型。在选择中，考虑将软件可靠性和花费信息作为基本原则，使软件包的平均可靠性达到最大值，它适合于包含几个软件的软件包。

对于开发者来说，可靠性和花费之间的关系是十分重要的。由于软件开发者是在有限资源（钱、时间）条件下进行工作，软件可靠性和花费又是两个相互关联而且竞争的量，因此两者间必须考虑一个折衷方案。

Noushin Ashrafi和Oded Berman给出的模型是假设为：

- 每个程序的可靠性和花费是明确已知的；
- 预算是有限的；
- 每个功能使用频率已知（由用户提供）。

下面我们简要介绍两种模型。

·模型1

采用拉格朗日松驰算法，为每个功能选择一个程序，以使平均可靠性达到最大，且购买软件的花费仍保持在预算之内。该模型不考虑程序的冗余，其可靠性计算公式为：

$$\max \overline{R}=\sum_{k=1}^{K} F_k R_k$$

其中约束条件为：

$$\sum_{j=1}^{m_k} x_{kj}=1 \qquad k=1\cdots K \qquad \text{(i)}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{m_k} x_{kj}C_{kj}\leqslant B \qquad \text{(ii)}$$

这里$x_{kj}=0,1 \quad k=1\cdot\cdot K;\ j=1\cdot\cdot m_k$

$$R_k\equiv\sum_{j=1}^{m_k} x_{kj}R_{kj}$$

k表示需要执行的软件包的功能数目；F_k为功能k使用的频率；m_k为功能k可得到的程序数；R_{kj}表示执行功能k的程序j的可靠性；x_{kj}为标志符：如果选择执行功能k的程序j则为1，反之为0；$\overline{R}$表示软件包的平均可靠性；C_{kj}为并发执行功能k的程序j的花费；B为可获得的预算。

·模型2

在考虑冗余的条件下，为每个功能选择一个以上的程序，按照一定的验收标准，从中选取最优程序组，从而在预算下使软件包的平均可靠性达到最优。

模型中采用动态编程算法，其可靠性公式为：

$$\max \overline{R}=\sum_{k=1}^{K} F_k R_k$$

其中约束条件为：

$$\sum_{j=1}^{m_k} x_{kj}\geqslant 1 \qquad k=1\cdot\cdot K \qquad \text{(i)}$$

$$\sum_{k=1}^{K}\sum_{j=1}^{m_k} x_{kj}C_{kj}\leqslant B \qquad \text{(ii)}$$

这里$x_{kj}=0,1 \quad k=1\cdot\cdot K;\ j=1\cdot\cdot m_k$

$$R_k=1-\prod_{j=1}^{m_k}(1-R_{kj})^{x_{kj}}$$

各项参数含义见模型1。

这两种模型中，模型2比模型1花费大，因此多用于故障引起后果较为严重的部门，如军队，急救中心等。

2.比较准则

在模型比较方面，Iannino、Musa、Okumoto、Littlewood曾给出了一些比较的准则[1]。准则的内容包括：有效性预测、效力、假设的质量、适用性及简单性。

·预测有效性。预测有效性是模型依据当前和过去故障行为预测未来故障行为的能力。这种能力只是在故障行为正在变化时是有意义的。两种通常用于预测有效性的方法是故障数方法和故障时间方法。故障数方法较故障时间方法更为实用。

·效力。效力指模型对软件管理者、工程师以及计划和管理软件开发的用户所需要的精确量的估计能力。这种能力的程度是由这些量的相对重要性决定的。

·假设的质量。要判断假设的好坏，可以通过测试的方法。如果测试较为困难，可通过逻辑一致性的观点和软件工程经验来估价是否合理。最后须看此假设是否清晰明了，是否可由它确定一个应用于特别环境的模型。

·适用性。一个模型可通过它对软件系统变化的规模、结构、功能等的适用程度来判定它在不同的开发环境、运行环境、生存周期是否可用。

·简单性。是指模型为确定参数而做的数据收集应是简单的；在概念上应是简单的；模型中参数量也是简单的。

以上五项准则对任意一个模型来说，并非要求同时满足，但在模型比较中，这五项准则基本上可以反映出可靠性模型的优劣和适用范围。

管理信息系统 MIS 数据流图 一致性 (8)

计算机科学1993 Vol.20№.6

47-50 TP11

MIS数据流图一致性检查的研究与实现

姚 俊 胡上序 冯树椿

(浙江大学计算与信息中心 杭州310027)

摘 要

Data Flow Diagram (DFD) is the core of the structured analysis method. In this paper, from the basic concept about DFD in the field of Management Information System (MIS), we define the consistency of DFD, and then mainly discuss the algorithms of DFD consistency check and the data structure for implementing them.

一、引 言

结构化分析设计方法是一种较为成熟和完善的MIS开发方法，它直观易学，普遍流行，并有不少已实现的系统正在正常运行。

尽管随着MIS的深入研究，结构化分析设计方法暴露了一些严重缺点，而提出一些新的MIS开发方法，如快速原型法和面向对象方法等，但结构化分析设计方法依旧保持着生命力，并且在新的开发方法中，往往被结合使用。

数据流图是从数据流向的角度来描述软件功能要求的一种方法，由于它具有直观、简洁等优点，已经为广大软件开发人员和软件用户所理解和接受，从而成为结构化分析的核心。它不仅表示了系统内部的数据流向和与外部联系的界面，而且表示了系统逻辑处理的功能。不论是现行系统或将由计算机来处理的新系统，其业务的逻辑关系都可以由数据流图来表示。

以往的数据流图都是由人用笔在纸上描绘的，这导致MIS开发人员要花费大量的精力和时间用于绘制数据流图，并且可修改性

姚俊 博士生，主要从事信息系统开发方法和工具、人工智能应用等方面的研究。胡上序 教授，博士导师，主要从事信息系统、神经网络、人工智能和数据处理等方面的研究。

四、结束语

软件可靠性研究和利用在我国还是一个尚未受到足够重视的领域。长期以来，我们的软件开发停留在个体手工业和小集体作坊作业规模上，许多软件开发过程并无质量监督人员跟踪记录，也不存在软件可靠性评估问题，这无疑对于我国软件行业的发展是不利的，应引起有关方面重视。

致谢 在本文的写作过程中，得到了赵致琢老师的悉心指导，特此感谢。

参考文献

[1] "Software Reliability", Anthony Iannino, John D. Musa, Advance in Computers, Vol.30 1990

[2] "Optimization Models for Selection of Programs, Considering Cost and Reliability", Noushin Ashrafi, Oded Berman IEEE Transaction on Reliability, Vol41, №2 1992