考虑故障相关的软件可靠性增长模型研究

赵 靖 张汝波 顾国昌

(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

摘 要 软件可靠性增长模型是用来评估和预测软件可靠性的重要工具.目前,绝大多数的软件可靠性增长模型并没有考虑故障之间的相关性,也没有考虑测试环境和运行环境的区别.文中提出了一种随机过程类非齐次泊松过程(NHPP)中的考虑故障相关性、测试环境和运行环境差别的模型.在两组失效数据上的实验分析表明:对这两组失效数据,文中提出的模型比其他一些非齐次泊松过程类模型的拟合效果和预测效果更好.

关键词 软件可靠性增长模型;故障相关性;测试环境;运行环境

中图法分类号 TP311

Study on Software Reliability Growth Model Considering Failure Dependency

ZHAO Jing ZHANG Ru-Bo GU Guo-Chang

(*School of Computer Science and Technology, Harbin Engineering University, Harbin* 150001)

Abstract Software reliability growth model (SRGM) is one of the important tools to evaluate and predict software reliability. However, software failure dependency, the differences of testing environment and operational environment are not considered by most SRGMs. This paper proposes a software reliability growth model that consider software failure dependency and differences of testing and operational environment. The experimental results based on two failure data sets show that the proposed model has better predictive power and curve fit than other NHPP SRGMs.

Keywords software reliability growth model; software failure dependency; testing environment; operational environment

1 引 言

随着计算机应用领域的不断拓展,软件的规模越来越大,结构和功能越来越复杂.人们对高质量软件的需求也更加迫切.软件可靠性是软件质量的重要指标之一,为了评估和预测软件产品的可靠性,一系列基于非齐次泊松过程(Non-Homogeneous Poisson Process, NHPP)的软件可靠性增长模型(Software Reliability Growth Model, SRGM)被相继提出[1].绝大多数软件可靠性增长模型都假设故障之间是相互独立的[1-4],实际上,在软件的测试过程中,故障之间的相关性是普遍存在的[5-8].例如,测试者为了寻找软件故障的根源,要使用一系列的测试手段和测试方法,如基于结构、集群和随机的测试.这些测试方法通常运行一系列的测试用例,这些测试用例之间具有一定的相关性.因此,为了准确地评估和预测软件可靠性,在建立软件可靠性增长模

收稿日期:2007-05-05;修改稿收到日期:2007-07-07.本课题得到哈尔滨工程大学基础研究基金(HEUST07021)、中国博士后基金、黑龙江省博士后基金资助.**赵 靖**,女,1973年生,讲师,主要研究领域为软件可靠性评估、容错计算、软件测试和智能计算.E-mail: zhaoj@hrbeu.edu.cn.**张汝波**,男,1963年生,教授,博士生导师,主要研究领域为智能机器人、机器学习和容错计算.**顾国昌**,1946年生,教授,博士生导师,主要研究方向为智能控制、智能机器人和机器人智能技术.

型时要考虑故障之间存在的相关性.但是,近几年只有少数的学者研究过具有故障相关性的软件可靠性增长模型.文献[5]将相关的故障分为两类,研究了这两类故障对软件可靠性的影响.文献[6]建立了考虑故障相关性的软件可靠性增长模型,并且研究了相关性对软件可靠性的影响.文献[7]在文献[6]的基础上,提出了故障相关性的3种形式,建立了考虑故障相关性的、基于Markov更新过程的软件可靠性增长模型.尽管文献[7]提出了故障相关性的软件可靠性增长模型,但是模型评估的关键数据,如评价两次运行之间的相关性参数是很难得到的,所以考虑故障相关性的软件可靠性增长模型,很难定量地计算软件可靠性.文献[8]在建立模型时,将软件故障之间的相关性分为两类,一类是独立的故障,另一类是相关的故障,对这两类故障进行分别建模.

现有的考虑故障相关性的软件可靠性增长模型对软件可靠性评估的精度与预测能力都有很大的提高,使模型的假设更贴近于实际.然而,在建模的过程中存在很大的局限性.第一,文献[8]在建立软件可靠性模型时,将软件的故障分为独立的和相关的两类故障.实际上,在软件测试过程中,当独立的故障被排除时,相关的故障就会被检测和排除,这时相关的故障就会变成独立的故障,从而故障之间的相关关系就会改变.因此,不能简单地将软件的故障分为独立的和相关的故障.第二,对故障相关性的分类只是以一种简单的非形式化的方式进行.然而,在实际情况下,故障之间的相关性是非常复杂的,因此有必要对故障相关性进行形式化描述.最后,随着测试时间的进行,故障之间的相关关系是不断改变的,这种变化使得独立故障和相关故障的比例关系也在不断变化.因此,故障出现率$e(t)$定义为在时刻t,每排除独立故障出现的相关故障数,是随时间变化的.因为软件的测试过程是可以跟踪的[9-10],可以根据上一个版本或者相似版本的测试数据,得到相关故障出现率的变化趋势,从而可以准确地预测未来发布版本软件的可靠性.

另一方面,软件的测试剖面与运行剖面是存在差别的[11-14],在软件的运行阶段,绝大多数相关性故障已经被排除,故障检测率体现了故障本身具有的下降的趋势.因此,这两个阶段故障检测率是不同的.

本文首先对软件故障相关性用有向图进行形式化的分析,然后分析随时间变化的相关故障出现率,最后建立既考虑故障相关性,又考虑测试剖面与运行剖面差别的软件可靠性增长模型.

2 基于有向图的故障相关性分析

文献[8]把故障分为两类:独立的故障和相关的故障.文献[15]用S形软件可靠性增长模型描述这两种相关性,假设独立的故障在不同的程序树路径上,只有独立的故障被排除后,相关的故障才能被排除.这种分类方法对软件可靠性建模有一定的指导作用,然而,软件故障之间的相关关系是很复杂的,这种分类方法并没有对故障之间的相关关系进行详尽的阐述.因此,应该用形式化的方法描述故障相关关系.我们把故障之间的相关关系分为三类:一对一,一对多,多对一.下面给出故障相关关系定义并讨论其性质.

定义1. 故障相关关系$R(i,j)$.设R是故障节点集上的二元关系.对于任一有序的故障节点(i,j),当且仅当只有当故障节点i被发现并且被正确排除后,j才有可能被排除,称$R(i,j)$为故障相关关系.

性质1. 故障相关关系传递性.如果$(j,k)\in R$并且$(i,j)\in R$,那么$(i,k)\in R$.

软件测试过程中,由于故障的相关关系,只有独立的故障被排除后,相关的故障才有可能被发现并且排除,相关的故障这种被发现并且排除的行为被称为暴漏.

定义2. 故障相关关系图.故障相关关系图是一个连通有向图$G(V,E)$,V是一系列的点用来表示故障节点集,$E\subseteq V\times V$表示有向边集,用来表示故障之间的相关关系.$|V|$表示故障相关关系图中的节点数.对任一个从点v_1到v_2的边e,$e\in E$,v_1称边e的起始节点,记作$IV(e)$,v_2称边e的终节点,记作$TV(e)$.对$\forall v\in V$,$Indegree(v)$记作故障节点v的入度,等于进入点v的有向边$\{e_1,e_2,\cdots,e_k\}$,$1\leq k\leq|V|-1$,当且仅当故障数等于$\{IV(e_i),1\leq i\leq|V|-1\}$的故障节点被排除后,故障$v$才能被排除.$Outdegree(v)$记作点$v$的出度,代表从故障节点$v$发出边数$\{e_1,e_2,\cdots,e_k\}$,$1\leq k\leq|V|-1$,只有故障节点$v$被发现并且排除后,故障节点等于$\{TV(e_i),1\leq i\leq|V|-1\}$的故障才可能被发现并且排除.故障相关关系图的一个路径是由一系列的边$(e_1,e_2,\cdots,e_k)$组成的,其中,$TV(e_i)=IV(e_{i+1})$,$1\leq i\leq k-1$.假设$v_l=IV(e_l)$,$1\leq l\leq k$并且$v_T=TV(e_k)$,称这个路径为从v_l到v_T的故障相关关系路径.

定义3. 直接相关.记作故障之间的直接相关

关系为 $D(i,j)$. 对于一个故障相关关系图 $G(V,E)$，如果 $(i,j)\in R$，并且如果 $\exists m_1,m_2,\cdots,m_k,k\leq|V|$ 使得 $(i,m_1),(m_1,m_2),\cdots,(m_k,j)\in E$，那么 $(i,j)\in D$. 一对一的相关关系记作 $OO(i;j)$. 如果 $(i,j)\in D$，当 i 被发现后，故障 j 暴漏并且当且仅当只有 j 被暴漏，那么 $(i,j)\in OO$；一对多的相关关系记作 $OM(i;j_1,j_2,\cdots,j_n)$. 如果 $(i,j)\in D$，当 i 被发现后，有一个以上的故障数 $j_1,j_2,\cdots,j_n$ 被暴漏，那么 $(i,j)\in OM$；多对一的相关关系记作 $MO(i_1,i_2,\cdots,i_n;j)$. 如果 $(i,j)\in D$，当且仅当一个以上的故障数 $i_1,i_2,\cdots,i_n$ 被发现，故障 j 才能被暴漏，那么 $(i,j)\in MO$.

一对一、一对多和多对一直接相关关系用有向图表示如下，图中用圈表示故障节点，相关关系用箭头指向表示.

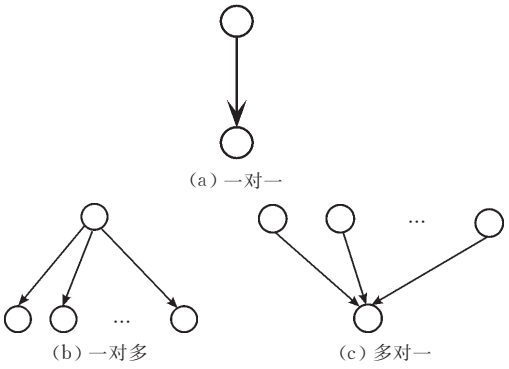

图 1　直接相关关系分类

定义 4.　独立的故障. v 被称为独立的故障当且仅当 $Indegree(v)$ 等于 0，$Outdegree(v)$ 表示与 v 相关的故障数.

定义 5.　多故障相关关系图. 由多个故障相关关系有向图构成一个不连通的多故障相关关系图. 一个多故障相关关系图是 $m+1$ 元组 $(V,E_1,E_2,\cdots,E_m)$，对所有的故障节点集 V 作出一个划分，把 V 分成非空的子集 $V_1,V_2,\cdots,V_m$，使得两个节点 v_j，v_k 是连通的，当且仅当它们属于同一个 V_i，对于 $\forall(V_i,E_i),1\leq i\leq m,(V_i,E_i)$ 是一个故障相关关系图.

定义 6.　软件测试得到的所有故障及其相关的故障构成一个多故障相关关系图.

定义 7.　初始故障数. 任意一个故障相关关系图 $G(V_i,E_i)$，对所有的 $v\in V_i$，$Indegree(v)$ 等于 0 的故障节点数记作 $NZ\text{-}Indegree(V_i,E_i)$. 对一个多故障相关关系图，初始的故障数用下式表示：$\sum_{i=1}^{m}NZ\text{-}Indegree(V_i,E_i)$.

定义 8.　相关系数. 相关系数是用来表示故障的相关程度，记作 $DC(LP)$. 对于每个故障相关关系图的所有路径，$LP(V_1,E_1)$ 表示故障相关关系图 $G(V_1,E_1)$ 最长的路径，对于一个多故障相关关系图，相关系数 $DC(LP)$ 被定义为平均的路径长度 $\sum_{i=1}^{n}\frac{1}{n}\cdot LP(V_i,E_i)$，也可以定义一个加权平均值 $\sum_{i=1}^{n}\frac{1}{n}\cdot\delta_i\cdot LP(V_i,E_i)$，其中，$\delta_i$ 代表一个故障相关关系图的权值.

软件测试的测试顺序可由下面的步骤执行. 对任意一个故障相关关系图 $G(V_i,E_i)$，除了一些不满足测试准则的不可达路径外，测试顺序就是相关关系的拓扑排序，对于多故障相关关系图，测试顺序就是组合优化问题. 组合优化的复杂度等于软件的故障节点数，因为每一个故障节点都只能被访问一次.

下面的例子用来解释故障相关性. 我们将每一个故障相关关系图称之为团. 软件的故障组成的一个多故障相关关系图如图 2 所示. 图 2(a) 表示一个故障节点数较少的小团，仅包含一个独立的故障. 图 2(b) 表示比较大的团，包含了比图 2(a) 复杂的相关关系的故障节点. 最大的团如图 2(b) 所示的相关关系数较大，既包含了独立的故障，又包含了相关关系的故障. 软件测试时，当且仅当独立的故障被发现并且被排除后，相关的故障才有可能被排除. 经典的软件可靠性增长模型都假设故障之间是独立的，对软件可靠性评估和预测的精度有失准确性.

从图 2 可以看到，在软件测试开始时，共存在 7 个故障，也就是说，初始的能够检测的软件故障为 7 个. 随着软件测试的进行，相关的故障不断地暴漏并且被排除，软件中包含的总故障数会不断增加. 经典

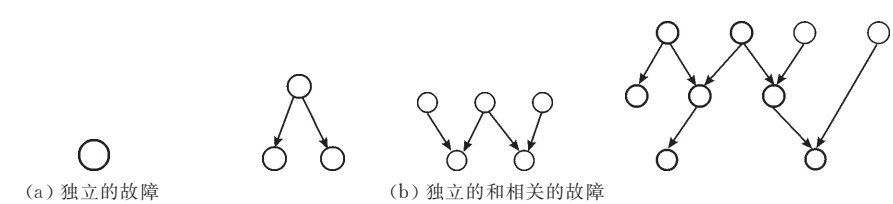

图 2　多故障相关关系有向图

软件可靠性增长模型的两参数指数模型由于其简单,因此被广泛使用,文献[16]在分析 NHPP 类具有指数形式的软件可靠性增长模型参数时,分析结果表明代表软件中包含最终故障数的参数,随着测试数据的增加,表现为递增的趋势,这种递增的趋势也可以由软件中故障之间存在的相关性来解释.

3 测试阶段考虑故障相关性的软件可靠性增长模型

根据故障的相关性关系,软件故障被划分为很多团,多个团就构成了一个多故障相关关系有向图,每个团中的故障数是不相同的.在软件测试过程中,包含故障数多的团,称为大团,对软件测试过程的影响要比小团大.软件测试的目的就是为了尽可能快地发现故障,对软件测试过程影响较大的故障首先被排除.一旦故障被发现,通常就会使用一系列的测试用例定位发生失效的根源[6],因此,排除大团故障的概率要高于排除小团的故障.其次,在大团中具有很多故障相关性的路径,如果每个路径被访问的概率相等,大团的故障被检测的概率要高于小团中故障被检测的概率.从另一方面看,软件测试人员对软件测试经验的逐渐积累,故障暴漏率高的输入首先被选取[5].因此,由于这两方面作用的结果,大团中的故障首先被排除,相关故障的出现率开始时比较高,随着测试的进行,大团将变成小团,故障的相关性会逐渐减弱,小团中的故障将会被检测并且排除,因此,相关故障出现率 $e(t)$ 会逐渐降低,这种下降的趋势可以用式(1)描述[17].

$$p_0(t) = A \cdot e^{-\beta t} \tag{1}$$

对式(1)作如下讨论:

(1)如果 $A=0$,那么 $p_0(t)=0$,表明在软件测试过程中故障之间是独立的,绝大多数 NHPP 类软件可靠性增长模型都是这个假设.

(2)如果 $A>0$,既有独立的故障又有相关的故障存在.

a)如果 $A<1$,那么 $p_0(t)<1$,故障相关性主要是多对一的关系.

b)如果 $A>1$,那么 $p_0(t)>1$,故障相关性主要是一对多的关系.

(3)β 的值表示下降的速率,β 的值越大,表明下降的速率越快,相关故障出现率变化的速率越快.

为了方便描述,本章中用到的一些符号及其含义解释如下:

$a(t)$:与时间相关的故障总数函数,包括已检测的故障数和未被检测的故障数;

b:测试阶段的故障检测率;

a:软件测试开始时能够被检测的故障数;

c:最终能够被检测的故障数;

b^o:运行阶段的故障检测率.

可以建立测试阶段故障相关性的软件可靠性增长模型,除了满足 NHPP 类模型的假设外,还要满足如下:

(1)一旦故障被发现,就会立刻被排除,不会引入新的故障;

(2)软件的故障被划分为独立的故障和相关的故障两类;

(3)相关故障出现率满足递减趋势.

模型的推导如下:

$$\frac{dm(t)}{dt} = b \cdot (a(t) - m(t)) \tag{2}$$

$$\frac{da(t)}{dt} = A \cdot e^{-\beta t} \frac{dm(t)}{dt} \tag{3}$$

式(2)与式(3)的初始条件为

$$m(0) = 0 \tag{4}$$

$$a(0) = a \tag{5}$$

软件测试阶段剩余故障数为

$$\chi(t) = a(t) - m(t) \tag{6}$$

对方程(6)的两边求导,易得

$$\frac{d\chi(t)}{dt} = \frac{da(t)}{dt} - \frac{dm(t)}{dt} = (A \cdot e^{-\beta t} - 1) \cdot \frac{dm(t)}{dt} \tag{7}$$

方程(7)也可以写成

$$\frac{d\chi(t)}{dt} = (A \cdot e^{-\beta t} - 1) \cdot b \cdot \chi(t) \tag{8}$$

方程(8)的初始条件为

$$\chi(0) = a(0) - m(0) = a \tag{9}$$

因此,由式(8)得到剩余故障数的期望值为

$$\chi(t) = a e^{-\int_0^t (1 - A e^{-\beta \tau}) \cdot b \, d\tau} \tag{10}$$

由式(2),测试阶段软件的失效强度为

$$\lambda(t) = m'(t) = b \cdot \chi(t) \tag{11}$$

因此,故障累计数均值函数推导如下

$$m(t) = \int_0^t \chi(u) \cdot b \, du = a \int_0^t b \cdot e^{-\int_0^u (1 - A e^{-\beta \tau}) \cdot b \, d\tau} \, du \tag{12}$$

经计算式(12)，易得

$$m(t) = a \cdot b \cdot e^{\frac{bA}{\beta}} \cdot \int_0^t e^{-b\left(u+\frac{A}{\beta}e^{-\beta u}\right)} du \quad (13)$$

设 $B(u)=e^{-b\left(u+\frac{A}{\beta}e^{-\beta u}\right)}$，$B^*(u)=\int_0^t e^{-b\left(u+\frac{A}{\beta}e^{-\beta u}\right)} du$，求近似值，得到

$$B^*(u) = \int_0^t e^{-b\left(u+\frac{A}{\beta}e^{-\beta u}\right)} du = \sum_{n=0}^{\infty} -\left(-\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b)} - 1) \quad (14)$$

式(14)代入式(13)，测试阶段的均值函数为

$$m(t) = a \cdot b \cdot e^{\frac{bA}{\beta}} \cdot \sum_{n=0}^{\infty} -\left(-\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b)} - 1) \quad (15)$$

故障总数为

$$a(t) = a\left(1 + \int_0^t A \cdot e^{-\beta u} \cdot b \cdot e^{-\int_0^u (1-Ae^{-\beta\tau}) b d\tau} du\right) \quad (16)$$

经计算，故障总数为

$$a(t) = a\left(1 + A \cdot b \cdot \sum_{n=0}^{\infty} -\left(-\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b+B)} - 1)\right) \quad (17)$$

4　测试与运行差别的考虑故障相关性的软件可靠性增长模型

在软件的测试阶段，相关的故障被不断地发现并排除，当所预测的指标达到用户的要求时，发布软件进入运行阶段. 在软件的运行阶段，故障检测是随机的，剩余的相关故障已经被降到最低，剩余的故障可以被看作是独立的[5].

运行阶段的软件可靠性增长模型的假设满足 NHPP 类模型 G-O 模型的假设，推导如下：

运行阶段的均值函数为 $m^o(t)$，推导如下：

$$\frac{dm^o(t)}{dt} = b^o \cdot [c - m(T) - m^o(t-T)] \quad t > T \quad (18)$$

其中，b^o 代表运行阶段的故障检测率

$$m^o(t) = (c - m(T))(1 - e^{-b^o(t-T)}) \quad t > T \quad (19)$$

因此，测试和运行阶段的软件可靠性增长模型为

$$m(t) = \begin{cases} a \cdot b \cdot e^{\frac{bA}{\beta}} \cdot \sum_{n=0}^{\infty} -\left(-\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b)} - 1), & t \le T \\ (c - m(T))(1 - e^{-b^o(t-T)}) + m(T), & t > T \end{cases} \quad (20)$$

式(5)～式(20)包含一个从 0～+∞的累计的表达式，很难用数值的方法计算，容易知道，当 $n \to \infty$，

$$\lim_{n\to\infty} -\left(\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b)} - 1) = 0 \quad (21)$$

因此，在满足式(20)和用户精度的要求下，从测试到运行考虑故障相关性的软件可靠性增长模型可以用下面的公式表示：

$$m(t) = \begin{cases} a \cdot b \cdot e^{\frac{bA}{\beta}} \cdot \sum_{n=0}^{M} -\left(-\frac{bA}{\beta}\right) \cdot \frac{1}{n!} \cdot \frac{1}{Bn+b} \cdot (e^{-t(Bn+b)} - 1), & t \le T \\ (c - m(T))(1 - e^{-b^o(t-T)}) + m(T) & t > T \end{cases} \quad (22)$$

因为这个模型既考虑故障相关性，又考虑测试剖面与运行剖面的差别，称为 TDO-SRGM.

4.1　比较标准

模型的拟和能力是使用误差平方和 *SSE* 和回归曲线方程的相关指数 *R-square* 来度量，这两个指标分别定义如下

$$SSE = \sum_{i=1}^{n} (y_i - \widehat{m(t_i)})^2 \quad (23)$$

$$R\text{-}square = \frac{\sum_{i=1}^{n} (\widehat{m(t_i)} - \bar{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (24)$$

其中 n 表示失效数据集中失效样本的数量，$\widehat{m(t_i)}$ 表示到 t_i 时刻为止故障累计数的估算值，y_i 表示到 t_i 时刻为止故障累计数的实测值.

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \quad (25)$$

SSE 的值越小，曲线拟合得越好；*R-square* 的值越接近于 1，曲线拟合得越好.

定义估计准确性为 *PE*，定义如下[3]：

$$PE = \left| \frac{m_a - c}{m_a} \right| \quad (26)$$

其中，m_a 代表软件测试结束后观测的实际失效数，c

代表累计故障数的估计值.

RE(Relative Error)用来计算模型预测能力[3]. *RE* 的定义如下:

$$RE = \frac{m(t_q) - q}{q} \tag{27}$$

假设在软件测试结束时 t_q 观测到 q 个软件故障数,使用失效数据 $t_e(t_e \leq t_q)$ 估测软件可靠性模型均值函数 $m(t)$ 的参数,然后将这些参数值代入到 $m(t)$ 中,可以得到软件测试结束时间 t_q 的累积失效数的估计值.得到的估计值与实际值进行比较,重复不同的失效时间间隔 t_e. 最后可以通过图形描述出 *RE* 值.

4.2 两个实例应用

为了检验 TDO-SRGM 模型的拟合效果,用最小二乘法,在现已发表的第一组数据[13]和第二组数据[18]上,分别应用 TDO-SRGM 模型、G-O 模型、Yamada 延迟 S 型、Logistic 模型、增强 G-O 模型[19]和 Huang 的相关性模型[8]进行实验.

4.2.1 第一组数据集

首先在第一组数据集上通过实验确定 M 的值.应用最小二乘法求解模型的参数,当 $M \geq 10$ 时,TDO-SRGM 的参数趋于稳定.求解的参数结果表示如下:初始的能够检测的故障数 a 为 133,最终能够检测的故障数为 207.3.软件已经经历了足够的测试.故障检测率 b 为 0.009226,$\beta = 0.05278$,$A = 94.42$,并且 $b^o = 0.000382$,表示相关故障出现率很高.

除此以外,表 1 也列出了其它的两个比较标准:*SSE*,*R-squares*,*PE*. 从表 1 可以看出,TDO-SRGM 的这两个比较标准值要优于其它的模型.

表 1 第一组数据集上模型的比较结果

比较标准	LSE	*SSE*	*R-square*	*PE*/%
TDO-SRGM	$c=207.3$ $a=133$ $b=0.009226$ $b^o=0.000382$ $A=94.42$ $\beta=0.05278$	413.3	0.9979	2.11
Huang 的相关性模型	$a=216.6$ $f=150.8$ $p=0.9724$ $r=0.08933$	636.3	0.9968	6.69
Yamada 延迟 S 型	$a=213.5$ $b=0.09348$	806.3	0.9962	5.17
G-O 模型	$a=301.3$ $b=0.02431$	3914	0.9814	48.42
增强 G-O 模型	$a=208$ $b=0.008258$ $c=1.543$	565.8	0.9973	2.46
Logistic 模型	$a=197.9$ $b=12.41$ $c=0.1419$	882.9	0.9958	2.51
Compertz 模型	$a=208.9$ $b=0.03101$ $c=0.9139$	434.6	0.9979	2.906

图 3 描述了 TDO-SRGM 与 Logistic 模型、Compertz 模型、Huang 的相关性模型的 *RE* 值. 从图 3 可以看出,TDO-SRGM 与这几个模型相比较,*RE* 值要比这几个模型更快地趋近于 0.

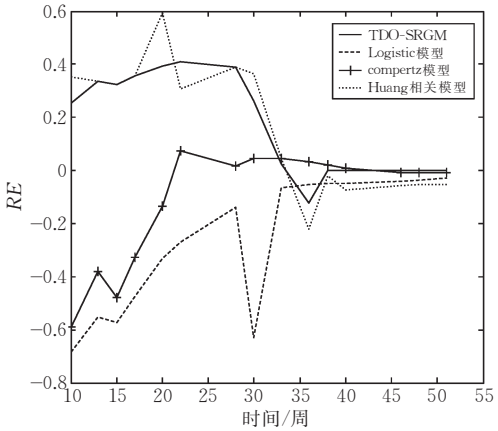

图 3 TDO-SRGM 与 Logistic 模型、Compertz 模型、Huang 相关模型 *RE* 值的比较

图 4 描述了 TDO-SRGM 与增强 G-O、G-O 模型、Yamada 延迟 S 型的 *RE* 值. 从图 4 可以看出,TDO-SRGM 与这几个模型相比较,*RE* 值要比这几个模型更快地趋近于 0.

参数敏感性分析的结果见表 2. 可以看到,c,a,b 与 b^o 对于估计的故障数有较大的影响,其余的参数对于估计结果的影响比较小.

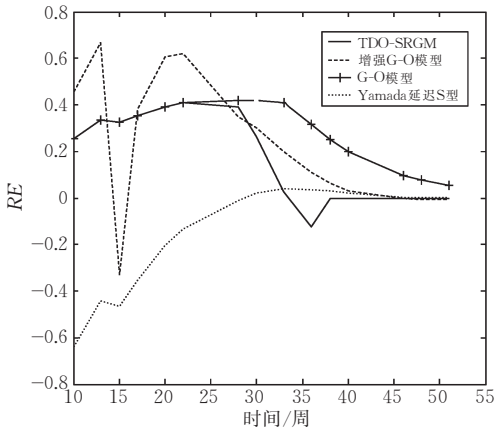

图 4 TDO-SRGM 与增强 G-O、G-O 模型、Yamada 延迟 S 型 *RE* 值的比较

参数 c 代表软件最终能检测的故障数,而且是评估软件可靠性的重要参数. 表 3 列出了本模型与其它模型对于这个参数的置信区间.

从以上比较的表格与图形可以得到下述结论:在第一组故障数据集上,TDO-SRGM 比其它几个模型拟合和预测能力要好.

表 2　参数敏感性分析的结果

x	c	β	A	b^0	b	a
20	0.15681	−0.03491	−0.00423	0.13651	0.16641	0.15376
15	0.12416	−0.02879	−0.00367	0.12461	0.15431	0.13416
10	0.07223	−0.04436	−0.03146	0.11486	0.14456	0.07153
5	0.00545	−0.03568	−0.04512	0.01079	0.00979	0.00568
−5	−0.09183	−0.05255	−0.05666	−0.08742	−0.07342	−0.09223
−10	−0.12457	−0.06718	−0.04567	−0.10996	−0.11196	−0.12467
−15	−0.18934	−0.08175	−0.04346	−0.12454	−0.13494	−0.18674
−20	−0.27208	−0.06897	−0.04306	−0.11785	−0.10845	−0.27338

表 3　参数 c 置信区间

比较结果	上限	下限
TDO-SRGM	221.8	194.1
Huang 的相关模型	258.6	195.2
Yamada 延迟 S 型	217.1	210
G-O 模型	335.5	267.2,
增强 G-O 模型	212.3	203.7
Logistic 模型	201.7	196.1
Compertz 模型	211.7	206.1

4.2.2　第二组数据集

第二组数据集上，由于所给的数据是标准化数据，TDO-SRGM 的最小二乘法求解的参数值并不代表实际意义. 因此，表 4 列出了其它的 3 个比较标准：*SSE*，*R-squares*，*PE*. 从表 4 可以看出，TDO-SRGM 的这 3 个比较标准值要优于其它的模型.

图 5 分别描述了所比较模型 *RE* 值，经过计算，在软件运行阶段结束时，TDO-SRGM、Chin-Yu Huang 式(20)模型、Yamada 延迟 S 型、G-O 模型和 Logistic 模型的 *RE* 值分别为−0.00046，−0.0375，−0.0369，−0.0264 和−0.0315. 因为增强 G-O 模型和 Logarithmic 模型很大地偏离了 0，并且 *RE* 值在运行阶段结束时分别为−0.0855 和−0.1069，因此在图 5 中我们没有给出这两个模型 *RE* 值的变化趋势.

表 4　第二组数据集上软件可靠性增长模型的比较结果

比较结果	*SSE*	*R-square*	*PE*/%
TDO-SRGM	0.0431117	0.9839	1.78
Huang 的相关模型	0.05303	0.9839	2.32
Yamada 延迟 S 型	0.2443	0.9685	3.69
G-O 模型	0.0644	0.9817	2.64
增强 G-O 模型	0.06285	0.9816	2.55
Logarithmic 模型	1.032	0.867	8.65
Logistic 模型	0.1191	0.9847	3.25

从以上比较的表格与图形可以得到下述结论：在第二组故障数据集上，TDO-SRGM 比其它几个模型拟合和预测能力要好.

因此，可以得到结论：在这两组故障数据集上，TDO-SRGM 模型的拟合与预测能力要比其它几个模型好.

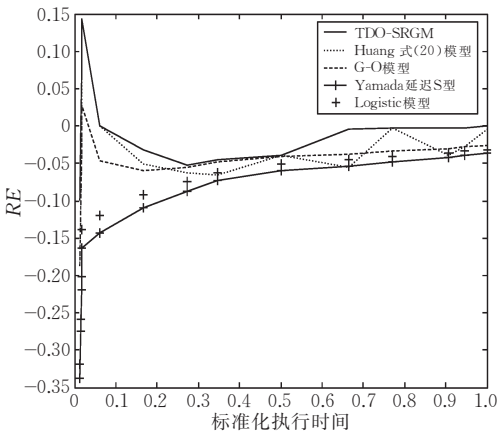

图 5　TDO-SRGM 与其它模型 *RE* 值比较

4.3　未来工作

下一步的工作将进一步具体分析影响故障相关性的因素，并且对这些影响软件测试过程的故障相关性因素进行量化分析，根据具体的工程，可以得到相关故障出现率的具体形式，将相关故障出现率融入到故障相关性的软件可靠性增长模型中，使得模型的假设更加合理，提高软件可靠性增长模型的评估和预测精度.

5　结　论

本文在有向图基础上，重点形式化分析了软件故障相关性，提出了相关故障出现率的概念，并且分析了随时间变化的相关故障出现率，最后建立了既考虑故障相关性，又考虑测试剖面与运行剖面差别的软件可靠性增长模型 TDO-SRGM. 实验结果表明，在已知的两组失效数据集上，TDO-SRGM 比其它的软件可靠性增长模型的拟合与预测能力更好.

参　考　文　献

[1] Lyu M R. Handbook of Software Reliability Engineering. New York: McGraw-Hill, 1996

[2] Yamada S, Osaki S. Software reliability growth modeling: Models and applications. IEEE Transactions on Software En-

gineering, 1985, 11(12): 1431-1437
[3] Musa J D, Iannino A, Okumoto K. Software Reliability: Measurement, Prediction, Application. New York: McGraw-Hill, 1989
[4] Miller D R. Exponential order static models of software reliability growth. IEEE Transactions on Software Engineering, 1986, 12(1): 12-24
[5] Wu Kang, Malaiya Y K. The effect of correlated faults on software reliability//Proceedings of the 4th International Symposium on software Reliability Engineering. Denver, co, 1993: 80-89
[6] Goseva-Popstojanova K, Trivedi K S. Failure correlation in software reliability models. IEEE Transactions on Reliability, 2000, 49(1): 37-48
[7] Dai Yuan-Shun, Xie Min, Poh Kim-Leng. Modeling and analysis of correlated software failures of mutiple types. IEEE Transactions on Reliability, 2005, 54(1): 100-106
[8] Huang Chin-Yu, Lin Chu-Ti, Kuo Sy-Yen, Lyu M R, Sue Chuan-Ching. Software reliability growth models incorporating fault dependency with various debugging time lags//Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04). Hong Kong, China, 2004: 186-191
[9] Musa J D. Software Reliability Engineering: Software, Faster Development and Testing. McGraw-Hill, USA, 1998
[10] Zou F Z. A change-point perspective on the software failure process. Software Testing, Verification and Reliability, 2003, 13(2): 85-93
[11] Zhao Jing, Liu Hong-Wei, Cui Gang, Yang Xiao-Zong. Research on the software reliability model considering differences between testing profile and operational profile. Journal of Computer Research and Development, 2006, 43(3): 503-508(in Chinese)
(赵靖,刘宏伟,崔刚,杨孝宗.考虑测试和运行差别的软件可靠性增长模型研究.计算机研究与发展, 2006, 43(3): 503-508)
[12] Zhao Jing, Liu Hong-Wei, Cui Gang, Yang Xiao-Zong. Software reliability model considering tesing environment and actual operational environment. Journal of Computer Research and Development, 2006, 43(5): 881-887(in Chinese)
(赵靖,刘宏伟,崔刚,杨孝宗.考虑测试环境和实际运行环境的软件可靠性增长模型.计算机研究与发展, 2006, 43(5): 881-887)
[13] Jeske D R, Zhang Xue-Mei, Pham L. Adjust software failure rates that are estimated from test data. IEEE Transactions on Reliability, 2005, 54(1): 107-114
[14] Zhang X, Pham H. Software field failure rate prediction before software deployment. The Journal of Systems and Software, 2006, 79(3): 291-300
[15] M. ohba. Infection S-shaped software reliability growth model. Stochastic Models in Reliability Theory. Berlin: Springer-Verlag, 1984: 142-162
[16] Malaiya Y K, Karunanityi N, Verma P. Predictability measures for software reliability models//Proceedings of the 1990 IEEE International Computer Software and Applications Conference(COMPSAC'90). Chicago, IL, 1990: 7-12
[17] Yamada S, M. ohba, Osaki S. S-shaped software reliability growth modeling for software error detection. IEEE Transactions on Reliability, 1983, 32(5): 475-484
[18] Teng Xiao-Lin, Pham H. A software cost model for quantifying the gain with considering of random field environments. IEEE Transactions on Computers, 2004, 53(3): 380-384
[19] Huang Chin-Yu, Lyu M R, Kuo Sy-Yen. A unified scheme of some nonhomogenous Poisson process models for software reliability estimation. IEEE Transactions on Software Engineering, 2003, 29(3): 261-269

ZHAO Jing, born in 1973, Ph. D.. Her research interests include software testing, software reliability evaluation, fault tolerance computing and wearable computing.

ZHANG Ru-Bo, born in 1963, professor, Ph. D. supervisor. His research interests intelligent robot, machine learning, fault tolerance computing.

GU Guo-Chang, born in 1946, professor, Ph. D. supervisor. His research interests include intelligent robot, machine learning, fault tolerance computing.

Background

NHPP (nonhomogeneous Poisson process) models, as a class of SRGMs, are extensively used. NHPP SRGMs have been quite successful tools in practical software reliability engineering. Many software reliability growth models (SRGMs) have been proposed, and most SRGMs assume that the successive software failure is independent and testing environment and operation environment is the same. However, the fault dependency is universal, and the differences of testing environment and operational environment exist in software testing. This paper incorporates fault dependency and differences of testing environment and operation environment. The relations of dependencies of fault can be classified into three types according to digraph theory, and many failure-node graphs can integrate into a multi-failure-node graph. Moreover, the decreasing emerging ratio of dependent faults is analyzed. Supported by the scientific Research Foundation of Harbin Engineering University (HEUST07021), the authors have studied the differences of testing environment and operational environment. This paper focuses on the evaluation accuracy and prediction precision of software reliability growth model incorporating fault dependency. The authors have worked on the evaluation algorithms for fault-tolerance and dependability. Some progress has been achieved. Furthermore, they will develop evaluation algorithm for dependability of very large scale network system.