双随机软件可靠性模型的建模

邹丰忠[1] 李传湘[2]

[1](武汉水利电力大学管理工程系 武汉 430072)
[2](武汉水利电力大学计算机科学系 武汉 430072)

摘要 本文在仔细分析了软件故障的物理过程及模型机理后,认为:由于软件故障过程的随机因素太复杂以致于用某一个传统随机过程不足以为软件故障过程建模.传统上采用的马尔可夫非齐次泊松过程模型或二项型式模型只能部分地表述软件的故障行为.若要全面地去表述软件的故障行为,就需要一个补偿随机过程.也就是说本文采用了两个随机函数来为软件故障行为建模.

关键词 软件,可靠性,模型,随机过程.

中图法分类号 TP311

软件可靠性模型理论是一门崭新的科学领域,它的研究对象就是软件系统及其行为过程,它大约始于70年代中期,随着计算机硬件成本的不断下降、软件规模和复杂度的上升,要求软件工程师们在日益紧缩的工期和预算条件下开发高质量软件系统的压力越来越大,人们不得不建立对软件性能和行为进行度量、预计和推断的理论.人们需要利用数学工具从量上来把握软件系统及其它因素(象质量、可靠性、使用剖面、测试方法、工具、工期、费用等).

在过去的20~25年里,软件可靠性模型理论的研究取得了巨大的成就,它提高了人们对软件开发、测试过程本质的认识,对软件生命周期的各个阶段从需求分析、设计、编码、组装、测试、运行到维护都有极高的价值.人们已看到了它无可限量的潜在效益.Motorola公司的Ralph Brettschmeider博士说:从已发放到市场的软件报告情况来看,软件可靠性模型理论的应用成百倍地改善了每千行代码的软件错误率.

自1973年IEEE计算机软件可靠性年会召开以后,软件可靠性模型理论就成为IEEE及其它学术机构、工业和政府部门的各种学术会议、学术著作和论文的主要研究题目之一.在模型理论的应用上,美国国防部(DoD)及Motorola公司已经作了明文规定和要求.但从总的情况来看,人们大多无意识地凭感觉来把握软件性能.这固然有人们的思想观念问题,也有理论研究跟不上的问题.事实上,理论与应用之间仍存在很大距离.主要是软件可靠性模型的精度不高、适应性不好.

本文提出了用两个随机函数建模的新思路,并从理论上和物理机理上证明了其可行性.然后率先使用时间序列分析理论结合随机分析理论建立了一个新模型.随后的模型实验、验证及比较工作证实,该模型较之传统模型在预测精度上有明显的改善.而且,从理论上来说,这个新模型适应性很好.

1 双随机软件可靠性模型的提出

曾经有学者提出这样的疑问:软件可靠度是什么?它存在吗?软件不存在磨损老化问题,如果软件中存在错误,那么它的可靠度是0%,如果软件中不存在错误,那么它的可靠度是100%.这种说法虽然偏颇,但能引起人们重新去思索软件可靠性模型的机理及其赖以生存的基础,静态地、孤立地去分析软件特性.上述说法不无道理.但动态地去分析,联系软件的运行环境等因素去考虑,情形就完全不同了.

软件可靠性模型赖以生存的根本依据是软件错误动态行为的不确定性.只问一个简单问题:人们为什么不能预计程序故障时间呢?如果知道了程序对于每一个可能输入的行为表现,如果能准确地预计将来的程序输入情况,那么就能完全精确地预计程序的故障时间.不幸的是,一般情况下,人们永远无法预先得到关于程序的全部信息.所以,软件故障过程确实是随机的,这就从根本上保证了软件可靠性模型存在的合理性.那么,随机性的准确源泉是什么?目前被

本文研究得到国家自然科学基金资助.作者邹丰忠,1962年生,讲师,主要研究领域为计算机软件理论及应用.李传湘,1935年生,教授,主要研究领域为计算机软件理论及应用.

本文通讯联系人:邹丰忠,武汉 430072,武汉水利电力大学管理工程系

本文1996-12-24收到原稿,1997-04-14收到修改稿

广泛采用的软件概念模型是输入输出模型.根据某些随机机制在输入空间I中选择一个点,经程序处理后,映射到输出空间O上的一个点,程序只不过是从I到O的一个映射而已,即P:I→O,只要程序接受的输入点属于引起软件故障的I_F子空间,那么程序的输出一定落在故障输出O_F子空间,也即程序经历了一次故障.输入点落入子空间I_F是随机的,所以用随机过程来建模是合理的.从输入空间的角度看,把选取一个程序输入点当作一次掷点实验,I_F与I的面积比以及使用剖面就决定了程序输入点落入I_F这一事件的概率.所以选取一个输入点,它是否落入I_F就意味着一次伯努利实验,当多次选取输入点时,它就成为了多重伯努利实验,事件发生的概率服从二项分布,所以用二项模型描述输入空间的掷点行为是合理的.而且当I_F很小时,用泊松过程更合适.

至此,本文要对传统的随机过程模型提出两点质疑:①传统模型总是假定故障过程一定服从某一个经典概率分布,比如说二项分布或泊松分布,这样做有什么道理?没有人给出理论上的证明.根据以上分析,传统模型这样做的结果是只考虑了输入的随机性,而忽略了测试过程的其它随机因素,实践证明这种做法不行.如果程序测试仅仅涉及上述软件概念模型,那么传统模型就是完全的.但实际软件是动态的,它一定要与人、计算机等因素交互作用.总之,传统模型遗漏掉了许多随机因素.这些随机因素是什么呢?测试工具,测试人员的思路、风格及经验,不完全排错,计算机环境因素,支撑系统(象操作系统、编译程序、数据库或网络等的特性),计算机的载荷、数据精度、数据记录误差,人们对故障定义理解上的分歧等.[1]也许还有其它的随机因素,但现已确知传统模型忽略了上述随机因素.这样,故障过程到底是否服从某一经典分布是不能肯定的,也就是说传统模型假定故障行为一定服从某一个经典分布是不合理的.②纵观传统模型,它们总是假定故障强度曲线按某一预定规律变化,比如指数衰减型、S型、Γ型、威布尔型、倒线性型、几何衰减型等等.硬性规定强度变化规律有什么依据呢?它合理吗?其实这些变化规律是从传统可靠性理论借鉴过来的,它们没有考虑计算机软件产品的特点.一件实体产品,譬如汽车、机床、晶体管等,受温度、湿度、磨损及老化的作用,按传统理论,它的故障强度曲线会按这些规律变化.但软件不同,它不会按某一人为预定的规律变化,而传统软件可靠性模型理论恰恰企图实际故障强度按某一预定规律去变化,比如GO模型认为故障强度是按指数衰减的,JM模型则认为故障强度是几何衰减的.这样,模型当然就不可能有很高的精度,只能碰运气.当某一工程项目的故障强度碰巧是指数型的或是几何衰减型的时候,它们的精度显得很好,一旦换一个工程项目,当它的故障强度不按这些规律变化时,这些模型的精度就很糟糕.本文认为这正是传统软件可靠性模型精度不高、适应性不好的根本原因.

综上所述,本文认为经典的随机过程模型不能完全刻画软件的故障行为.但经典模型有着雄厚的理论及实践上的根基,不宜轻易抛弃.所以本文建议在经典模型的基础上,辅助一个补偿随机过程,用两个互补的随机过程来为软件故障行为建模.补偿随机过程要能恰好弥补经典模型的不足,而又不蹈经典模型的覆辙,即不能人为规定它的分布和强度曲线.

2 随机函数的分解及复合叠加定理

第1节从物理机理上论证了传统随机过程不能全面地表征软件测试行为过程,并提出了辅之以一个补偿随机过程,用两个互补的随机函数为软件故障行为建模的设想.这就意味着将软件的动态故障行为分解成两个随机函数复合叠加的形式:一个是传统的马尔可夫非齐次泊松或二项形式随机函数,另一个是辅助补偿随机函数.然而这个设想在理论上成立吗?也就是说,这样的分解与复合存在吗?如果这个分解与复合存在,那么这个补偿随机函数的形式是怎样的呢?这正是本节要讨论的主要问题.下面在一般理论意义上,用构造法来证明随机函数分解与复合叠加的存在性,说明补偿原理,同时也就给出了补偿随机函数的一般结构形式.

定理. 如果给定随机函数$\{U(t)\}$和$\{N(t)\}$,那么,总存在$\{N(t)\}$的一个补函数$\{X(t)\}$,使得$\{U(t)\}$可以分解成$\{N(t)\}$与$\{X(t)\}$的复合叠加的形式.

证明,对于参数$t\in T$,T是某数集,设$\{U(t)\}$的概率空间为(Ω_u,F_u,P_u),由F_u和P_u总可以确定随机变量$U(t)$的一个概率密度函数$f_t(u)$,又设$\{N(t)\}$的概率空间为(Ω_n,F_n,P_n),同样道理,存在随机变量$N(t)$的密度函数$f_t(n)$,若它们相关,设其联合密度为$f_t(u,n)$

① 仍取$t\in T$

② 令随机变量$X(t)=U(t)-N(t)$,即　　$U(t)=N(t)-X(t)$　　(1)

规定F_x为所有满足下列双射关系的函数$X(t)$的集合

$$F_u\times F_n\xrightarrow{X(t)=U(t)-N(t)}F_x \tag{2}$$

③ 规定Ω_x为所有关于t的函数的集合,显然$F_x\subset\Omega_x$

④ 对任意给定的 $t\in[0,T]$,定义随机变量 $X(t)$ 的分布函数为

$$F_t(x)=P\{X(t)\leqslant x\}=\iint_{U(t)-N(t)\leqslant x} f_t(u)\cdot f_t(n)dudn \tag{3}$$

$$F_t(x)=P\{X(t)\leqslant x\}=\iint_{U(t)-N(t)\leqslant x} f_t(u,n)dudn \tag{4}$$

则 $X(t)$ 的密度函数为 $f_t(x)=\dfrac{dF_t(x)}{dx}$,根据 $X(t)$ 的密度函数,对 F_x 中的每一个具体函数 $x(t)$,规定其联合概率为

$$P_{x(t)}=P\{X(t_1)=x(t_1),\ldots,X(t_n)=x(t_n),\ldots\}\quad\forall\, t_i\in T,i=1,2,\ldots \tag{5}$$

这样对 F_x 中的任一具体函数 $x(t)$,$P_{X(t)}$ 就是它可能成为一次现实的概率.

再规定 $P_x=\{P_{X(t)}\,|\,x(t)\in F_x\}$,显见,$(\Omega_x,F_x,P_x)$ 定义了一个新的概率空间和一个新的随机函数 $\{X(t)\}$,并且随机函数 $\{U(t)\}$ 能分解为随机函数 $\{N(t)\}$ 与 $\{X(t)\}$ 的复合叠加. □

如果用 λ_u 表示 $\{U(t)\}$ 的强度函数,用 λ_n 表示 $\{N(t)\}$ 的强度函数,用 λ_x 表示 $\{X(t)\}$ 的强度函数,那么根据式(1),它们的均值函数满足

$$EU(t)=EN(T)+EX(t)\qquad t\in[0,T] \tag{6}$$

两边进行微分,得到它们的强度函数满足

$$\lambda_u=\lambda_n+\lambda_x \tag{7}$$

现在来考虑分解与复合叠加定理在软件可靠性模型建模中的应用.

如果用定义在 (Ω_u,F_u,P_u) 概率空间上的随机过程 $\{U(t)\}$ 表征被观察的实际软件故障行为,用定义在 (Ω_n,F_n,P_n) 概率空间上的随机过程 $\{N(t)\}$ 来表征传统模型对被观察软件故障行为的模拟,那么根据随机函数的分解与复合叠加定理,总可以找到一个定义在 (Ω_x,F_x,P_x) 上的随机过程 $\{X(t)\}$,使得随机过程 $\{U(t)\}$ 可以用 $\{N(t)\}$ 与 $\{X(t)\}$ 的复合叠加来表征.它的物理意义是:当模型 $\{N(t)\}$ 没有完全刻画实际软件故障行为时,或者说当模型 $\{N(t)\}$ 与实际故障行为不相吻合时,随机过程 $\{X(t)\}$ 就表征了这个不相吻合部分,而它恰好是被传统模型所忽略了的随机因素综合作用的结果.亦即传统模型 $\{N(t)\}$ 与补偿随机过程 $\{X(t)\}$ 复合叠加就全面地表征了实际软件故障行为过程 $\{U(t)\}$,这就是补偿机理.

下面给出本文讨论的双随机软件可靠性模型的表示式.

$$\{U(t)\}=\{N(t)\}\oplus\{X(t)\} \tag{8}$$

其中⊕表示对任意给定的 t 时随机变量的普通代数和,$\{N(t)\}$ 是本文第 1 节讨论的随机过程模型,$\{X(t)\}$ 是随机过程的特殊形式——时间序列.关于它的具体构造形式和分析将在后面讨论.

3 双随机软件可靠性模型的实现

首先要估计出经典模型的参数,这里采用极大似然法,设在 $t_1,\ldots,t_{m_e}$ 各时刻共观察到 m_e 个软件故障,似然函数为

$$ML(\beta;t_1,\ldots,t_{m_e})=f(t_1,\ldots,t_{m_e})P\{T_{m_e+1}>t_e\,|\,T_1=t_1,\ldots,T_{m_e}=t_{m_e}\}=\prod_{i=1}^{m_e}\lambda(t_i)]e^{-\mu(t)} \tag{9}$$

取对数,并对待估参数微商后令其等于 0,就有两个方程

$$\hat{\omega}_0=\frac{m_e}{1-(1+\hat{\Phi}t_e)e^{-\mu(t)}} \tag{10}$$

$$\frac{2m_e}{\hat{\Phi}}=\sum_{i=1}^{m_e}+\frac{m_e\hat{\Phi}t_ee^{\hat{\Phi}}}{1\quad(1+\hat{\Phi}t_e)e^{-\hat{\Phi}t_e}} \tag{11}$$

解方程得到经典模型的参数 $\hat{\omega}_0$ 和 $\hat{\Phi}$,这样就得到了传统模型 $\{N(t)\}$,又已观察到实际故障过程 $\{U(t)\}$ 的样本.应用分解及复合定理就可得到补偿过程 $\{X(n),n=0,\ldots,N\}$,剩下的工作是要分析补偿随机过程的统计特性.这里考虑使用时间序列分析理论,正如数理统计是分析随机变量统计特性的工具一样,时间序列分析理论是分析随机过程统计特性的工具.它有两个特点①对被分析的样本数据没有任何统计上的先决条件;②它能突出样本数据特色.考虑 $X(n)$ 取下列形式

$$\begin{aligned}&X(n)=m(n)+Y(n)\\&E[Y(n)]=0,E[Y(n)\overline{Y(m)}]=B(n-m)\end{aligned} \tag{12}$$

其中 $m(n)$ 为非随机序列,是 $X(n)$ 的均值部分.$Y(n)$ 是 $X(n)$ 的随机变化部分,根据样本数据,我们可以用非线性回归

的方式来估计 $m(n)$，令

$$m(n)=\sum_{\nu=1}^{t} a_\nu \varphi_\nu(n) \tag{13}$$

其中 $\varphi_i(n)=e^{i\lambda_i n}$，用最小方差估计

$$\sum_{n=1}^{N} |X(n)-\sum_{\nu=1}^{t} a_\nu \varphi_\nu(n)|^2 = \min \tag{14}$$

对上式求关于 a_ν 的偏导数并令它等于 0，就得到线性方程组

$$\sum_{j=1}^{t}\left(\sum_{n=1}^{N} \overline{\varphi_i(n)}\varphi_j(n)\right) = \sum_{n=1}^{N} \overline{\varphi_i(n)}X(n) \tag{15}$$

为方便起见，令

$$X=\begin{bmatrix} x(1) \\ \vdots \\ x(n) \end{bmatrix} \quad Y=\begin{bmatrix} y(1) \\ \vdots \\ y(n) \end{bmatrix} \quad \alpha=\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_t \end{bmatrix} \quad \Phi=\begin{bmatrix} \varphi_1(1) & \cdots & \varphi_t(1) \\ \cdots & \cdots & \cdots \\ \varphi_1(N) & \cdots & \varphi_t(N) \end{bmatrix}$$

那么 $\hat{\alpha}=(\Phi^T\Phi)^{-1}\Phi^T X$ 是无偏估计.

对角频率 λ_i 估计的依据仍然是使方差最小，即

$$(X-\Phi\alpha)^T(X-\Phi\alpha)=\min \tag{16}$$

整理后得到

$$I_{ZN}(\lambda)=\frac{1}{2\pi N}\left|\sum_{n=1}^{N} e^{-in\lambda}X(n)\right|^2 \triangleq 0 \tag{17}$$

式(17)就是周期图极大估计函数，利用该函数就可以估计出 λ_i. 在 $\hat{m}(n)$ 估计出来以后，就可求得余差序列 $\{Y(n)\}$，如果它是广义平稳的，就可直接处理，否则，还要对其进行一阶、二阶或三阶算子变换，使它成为一个广义平稳序列. 为了对余差序列进行滤波和预测，先要估计它的协方差函数 $B(\nu)$

$$\hat{B}(\nu)=\frac{1}{N}\sum_{n=1}^{N-|\nu|} Y(n+|\nu|)\,\overline{Y(n)} \tag{18}$$

为了进行滤波与预测，根据自回归 AR 模型，用 $Y(n)$ 的线性组合 $\sum_{i=1}^{N} C_i Y(S_i)$ 就能满足要求，解下列线性方程组，就可得到 C_i.

$$E\{\sum_{i=1}^{N} C_i Y(S_i)\overline{Y(n_j)}\}=E\{Y(n_0)\overline{X(n_j)}\} \tag{19}$$

这样就解决了余差序列的滤波与预测问题，从而也就能预测将来的故障 $\hat{X}(t)$.

为了计算其它软件性能度量值，我们还要利用经典模型，根据补偿随机过程模型的预测值来校正经典模型的参数，令

$$\mu(t)=\hat{\mu}(t)+\hat{X}(t) \tag{20}$$

其中 $\mu(t)$ 是实际值，$\hat{\mu}(t)$ 是传统模型的预测值，$\hat{X}(t)$ 是补偿模型的预测值.

$$\hat{\omega}_0[1-(1+\tilde{\Phi}t_n)e^{-\tilde{\Phi}t_n}=\hat{X}(n)+\hat{\mu}(t) \tag{21}$$

从上式中解出的 $\tilde{\Phi}$ 是校正过的参数，把它代入传统随机过程模型，就可按传统模型方法计算各种软件性能度量值.

4 模型的验证与比较

模型验证的依据是 Musa 的标准数据集.[2] 这些公开发表的数据集是从实际工程项目中收集来的，经过分析整理，保证其有很高的精度，专为模型比较、验证而设计.

软件可靠性模型所要解决的主要问题是软件性能预计. 即使人们想要知道的是软件当前的特性，实际上也是着眼于未来. 例如，当前软件故障率、软件完成某一工作而不发生故障的概率、下次故障前平均时间等等，实际上都是在预计未来.

讨论软件可靠性模型的精度实际上是讨论它预计及推断的准确度. 这个事实常为人们所忽略，人们往往会以一个模型能精确地解释软件测试过程的历史来证明该模型是有效的，然而能准确描述过去并不意味着能精确地预计未来.

为了进行模型验证，我们开发了双随机模型的实现程序，包括超越方程的求解、高阶线性方程组的求解、周期图频谱分析、线性相关性分析、相关性消解等几大模块. 下面的验证结果是基于 Musa 的 $T1$ 数据集，在假定已发生 60 个软件故障的情况下，运行模型程序得到的结果如表 1.

表1　补偿过程时间序列

序号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
差值	6.86	4.62	5.60	3.96	4.79	3.45	5.76	5.60	3.81	8.53	9.00	8.38	7.87	6.85	5.24	3.61
序号	17	18	19	20	21	22	23	24	25	25	27	28	29	30	31	32
差值	3.69	2.44	2.65	1.61	2.29	0.63	-0.5	0.28	2.02	2.27	1.70	1.42	-0.44	1.30	2.7	6.31

表1的差值就是补偿过程的一个样本，这个补偿过程时间序列的样本就是传统GO模型在各个点向前预测的结果与实际观察值的差值.根据以前的分析，这个补偿随机过程反映的是被传统模型所忽略了的各随机因素的综合作用效果.用计算机程序求出该补偿随机过程的统计结构后，再联合传统GO模型，就可进行模型参数的预测，从而对双随机模型进行验证.

验证结果如图1所示.可以看出，双随机模型的预测数据分布在实际数据邻近的两侧，而GO模型的预测数据偏离实际数据很远.上述结果说明，双随机模型的预测精度较之传统模型有明显的改善.另外，由于不对软件故障过程的概率分布、强度曲线作任何假设，从理论上讲，双随机模型的适应性很好.

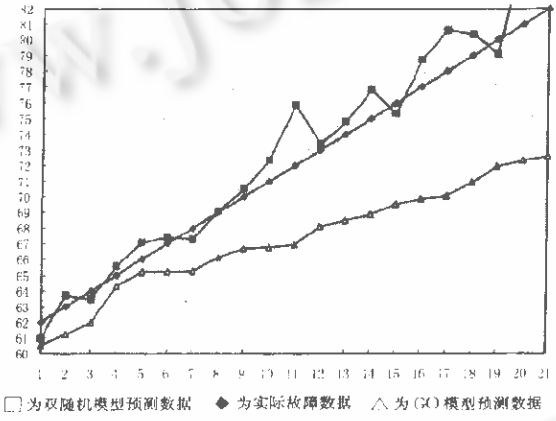

图1　GO模型和双随机模型预测结果比较

参考文献

1　Iyer Ravishanker K. Effect of system workload on operating system reliability: a study on IBM 3081. IEEE Transactions on Software Engineering, 1985, SE-11(12): 1438~1447

2　Musa J D *et al*. Software reliability, measurement, prediction, application. Mc Graw-Hill, 1987

A Dual Stochastic Function Model for Software Reliability

ZOU Feng-zhong[1]　LI Chuan-xiang[2]

[1](*Department of Management Engineering　Wuhan University of Hydraulic and Electric Engineering　Wuhan*　430072)

[2](*Department of Computer Science　Wuhan University of Hydraulic and Electric Engineering　Wuhan*　430072)

Abstract　After careful examination of the fundamental mechanics of software failure process, the authors suggested that the random factors in software failure process are too complex to be completely modeled on a single conventional stochastic process model. The generally used Markov or Non-Homogeneous-Poison-Process can partly explain the failure behavior only. To completely explain, a complementary stochastic process is needed. That is to say, two stochastic functions were suggested for modeling the failure behavior of software, hence the name of the model. This thesis presented the work on building such a model.

Key words　Software, reliability, models, stochastic processes.

Class number　TP311