硬软件交互的可靠性建模及其应用

胡慧芳,沈元隆

(南京邮电大学 电子科学与工程学院,江苏 南京 210003)

摘 要:开发了一种统一的可靠性模型,该模型对系统的硬件失效、软件失效和软硬件交互失效都可以作出解释。硬软件失效可以由熟知的建模方法来解决。而提出了一套利用马尔可夫过程来捕获硬软件交互失效的建模方法论,通过将其应用到真实的通信系统来说明该硬软件混合的建模方法。

关键词:可靠性建模;软件失效;硬件失效;软硬件交互失效

中图分类号:TP311 **文献标识码**:A **文章编号**:1674-7720(2011)05-0064-04

Reliability modeling of hardware and software interactions and its applications

Hu Huifang, Shen Yuanlong

(College of Electronic Science and Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: This paper develops an unified reliable model that accounts for failures in all three categories which are hardware failures, software failures and hardware-software interaction failures. Hardware and software failures are accounted for with well-known modeling approaches. This paper proposes a methodology using Markov processes to capture hardware-software interaction failures. It illustrates the combined hardware & software modeling approach by applying it to a real telecommunication system.

Key words: reliability modeling; software failures; hardware failures; hardware-software interaction failures

针对硬件可靠性和软件可靠性研究领域虽然有着大量的文献,但多数都只局限于单独的硬件或者软件子系统的研究。少数研究者设法为整个系统建立一个混合的可靠性模型,既包括硬件也包括软件。但他们通常认为硬件和软件子系统是相互独立的。

事实上,在许多现代计算机系统内,硬件和软件间的影响很大。例如,参考文献[1]对斯坦福大学的MVS/SP操作系统进行的软件错误分析发现,将近35%的软件错误是和硬件有关的。在这些系统中,硬件和软件子系统不是相互独立的,系统的可靠性受软硬件交互作用的影响。硬件和软件系统之间的交互作用表现在两个方面:积极的和消极的。参考文献[1]中提到的哈勃望远镜机械上的问题,经软件变更而得到了缓和。这意味着通过软件可以修正硬件问题,这就是软件所起的积极作用。另一方面,预料不到的硬件失效模型致使软件在一个非典型的操作模型下执行,而软件因没有被严格的测试过,结果导致系统的性能下降。在失效状态下,操作会更不可靠。

1 混合可靠性模型

1.1 基本假设

以下是混合可靠性模型的一些基本的假设

(1)无论是永久总体硬件失效、纯软件失效还是硬软件交互失效发生了,整个系统都失效。

(2)硬件失效可以模型化为韦伯模型,它的可靠性函数为 $R_{HW}(t)=e^{-\lambda t^{\beta}}$。

(3)软件失效可以模型化为NHPP模型,服从均值为 $m(t)$,给定持续时间 t 和软件启动时间 T,软件可靠度为:$R_{HW}(t)=e^{-(m(t+T)-m(t))}$。

(4)硬软件交互失效、硬件失效和软件失效是彼此独立的。

(5)硬件部件的失效率为 λ_1。

(6)局部硬件失效能以概率 P_1 被检测到,一旦检测到,就可通过软件方法以概率 P_2 得到恢复。虽然软件方法及相关的硬件可能找不到退化处,但是也不会有不正确的保护。

(7)一个检测不到的硬件退化可能会导致硬软件交

互失效(失效率为 λ_3),检测到的硬件退化可导致执行中止(失效率为 λ_4)。

(8)一旦局部硬件失效被检测到,失效的硬件部件就被修正或者是替换,通过软件恢复的概率为 μ_{1a},没有通过软件恢复的概率为 μ_{1b}。

如果退化被检测到却没有通过软件的方法恢复,局部硬件失效可能进一步以概率 λ_{2a} 演变为总体失效;如果退化被检测到并且通过软件的方法恢复,则以概率 λ_{2b} 演变为总体失效;如果退化没有被检测到,则以概率为 λ_{2c} 演变为总体失效。

1.2 模型

基于(1)和(4)的假设,整个系统的可靠度为:

$$R_c(t)=P_r\{t\text{ 时刻无总体失效}\}\times P_r\{t\text{ 时刻无纯软件失效}\}\times P_r\{t\text{ 时刻无硬软件交互的失效}\}=R_{HW}(t)R_{SW}(t)R_{HW/SW}(t)\quad(1)$$

式中,$R_{HW}(t)$ 和 $R_{SW}(t)$ 是纯硬件可靠性函数,纯软件可靠性函数由假设(2)和(3)分别决定的,而软硬件交互失效可靠性函数 $R_{HW/SW}(t)$ 来自于假设(5)~(9)。

图 1 是 HW/SW 失效模型的状态转移图。

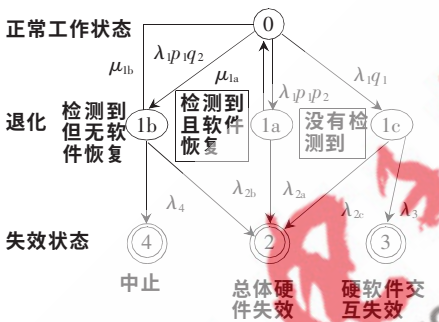

图 1 HW/SW 失效模型状态转移图

用 $Q_j(t)$ 表示 t 时刻处于状态 j 的概率。基于马尔可夫过程给出以下微分方程:

$$Q'_0(t)=-\lambda Q_0(t)+\mu_{1a}Q_{1a}(t)+\mu_{1b}Q_{1b}(t)$$
$$Q'_{1a}(t)=-\lambda_1 p_1 p_2 Q_0(t)-(\mu_{1a}+\lambda_{2a})Q_{1a}(t)$$
$$Q'_{1b}(t)=\lambda_1 p_1 p_2 Q_0(t)-(\mu_{1b}+\lambda_{2b}+\lambda_4)Q_{1b}(t)$$
$$Q'_{1c}(t)=\lambda_1 q_1 Q_0(t)-(\lambda_{2c}+\lambda_3)Q_{1c}(t)$$
$$Q'_2(t)=\lambda_{2a}Q_{1a}(t)+\lambda_{2b}Q_{1b}(t)+\lambda_{2c}Q_{1c}(t)$$
$$Q'_3(t)=\lambda_3 Q_{1c}(t)$$
$$Q'_4(t)=\lambda_4 Q_{1b}(t)$$

在这些系统状态中,状态 2、3 和 4 是失效状态,而状态 1a、1b 和 1c 是退化工作状态,状态 0 是全工作状态。如果初始状态 $Q_0(0)=1$,则可以得到附录中微分方程式的解。到 t 时刻无硬软件交互失效的概率为:

$$R_{HW/SW}(t)=P_r\{t\text{ 时刻无永久硬软件交互失效}\}=Q_0(t)+Q_{1a}(t)+Q_{1c}(t)\quad(2)$$

由此可以得到式(1)中总体系统失效率为:

$$R_c(t)=R_{HW}(t)R_{SW}(t|T)R_{HW/SW}(t)=e^{-\lambda\beta}e^{-m(t+T)-m(t))}\times Q_0(t)+Q_{1a}(t)+Q_{1c}(t)\quad(3)$$

下面通过其在通信系统中的应用来阐明硬软件混合模型方法。

2 电信应用

2.1 系统描述

通过将模型应用到无线 BSS 的一个特殊部件来说明。这个特殊部件是服务器,它是 BSc 的一部分。服务器按 N+1 组配置的,组操作在负载均衡的环境下进行的。当一个服务器失效,组中剩下的服务器接管它所执行的任务。每个服务器上运行的故障恢复软件负责管理恢复进程。

表 1 列出占用时间和失效数据。表中所示的 6 个月中的每个月,总计硬件的占用时间的系统日期,不管硬件运行的是软件 R1(2 列)还是 R2(4 列),都已给出。第 3 和 5 列是服务器的应用编号,是通过把总的占用时间按每月的天数划分而得到的。硬件设计也是一样,不考虑它执行的是哪个软件,第 6 列表明了每月来自所有的服务器总的硬件失效的数目。第 7 列说明了服务器只运行软件 R2 时的软件失效数。

表 1 该电信实例中的占用时间及失效数据

月份	服务器运行软件 R1 的占用时间/s	软件 R2 在服务器上的应用编号	服务器运行软件 R2 的占用时间/s	软件 R1 在服务器上的应用编号	所有服务器中的硬件失效数	软件 R2 的失效数
1	8 882	31	961	287	23	4
2	6 120	139	4 170	204	32	1
3	2 465	284	8 804	80	32	5
4	527	383	11 873	17	21	4
5	45	437	13 110	2	44	3
6	0	458	14 198	0	55	1
总计	18 039		53 116		207	18

当区域中有软件失效时,用户会通知技术支持人员,由技术员创建故障列表。如果故障的性质被诊断是程序上的错误,通常会关闭故障列表,而当前的软件释放令不改变。如果故障已经确定与软件失效有关,则要为故障分配安全级别。区域中首次高安全性级别故障的发生与软件更新之间的间隔通常不会超过两周。如果域中有异常多点故障,有关这个重复事件的列表被认为是源故障列表的再现。在绘制表 1 时,只有源故障列表对高安全性级别的故障有反映。

表 1 没有对硬软件交互失效做出详尽的说明。分析故障列表的根本原因时,没有对其进行硬软件交互失效的分类。多数情况下,域中发生的硬软件交互失效都被划分为了硬件失效,因为失效中的硬件因素在故障列表被创建时比软件因素更明显。下面,在调查硬软件交互失效的影响时,在不确定的假设条件下分析硬件失效实际为硬软件交互失效的概率 F 有多大。

2.2 配置模型

为了安装硬软件交互失效模型,只关注运行 R2 的

服务器，因为硬软件交互失效模型包含软件失效率 λ_3，这个参数依赖于软件排错。由于这个实例中系统没有内置的检测硬件退化的机制，在表 1 中，令 $P_1=0$。另外，为了满足 $\lambda_3>>\lambda_{2c}$，则进行进一步简化，令 $\lambda_{2c}=0$，则由式(2)可以导出：

$$R_{HW/SW}(t)=(\lambda_3 e^{-\lambda_1 t}-\lambda_1 e^{-\lambda_3 t})/(\lambda_3-\lambda_1) \quad (4)$$

将特定服务器的 HW/SW 失效次数模型化为由式(4)给出的带有到达间隔分布的更新进程。用 T_i 和 X_i 分别表示服务器的总计占用时间(系统日期)和 HW/SW 失效次数，这些服务器是第 i 个月初部署在域中的。由更新定理在更新进程时指出：假如 T_i 很大，X_i 可能的期望值为 T_i/μ，此处 $\mu=1/\lambda_1+1/\lambda_3$。$(\lambda_1,\lambda_3)$的估值可以由非线性最小二乘法求最小值：$SSE(\lambda_1,\lambda_3)=\sum_{i=1}^{6}[X_i-T_i/(1/\lambda_1+1/\lambda_3)]^2$。但还有两点需要声明。

(1)在估计过程中，在无附加信息条件下，(λ_1,λ_3)不是唯一确定的。将(λ_1,λ_3)参数化为(G,λ_3)，其中 $G=\lambda_3/\lambda_1$，G 被认为是已知量，且是随着精度研究而变化的。在应用中，希望 $G>>1$。利用新的参数，视 G 为已知，λ_1 的最小二乘估计值 $\hat{\lambda}_1=(1+1/G)(\sum_{i=1}^{6}X_iT_i)/\sum_{i=1}^{6}T_i^2$，然后得到$\hat{\lambda}_3=G\hat{\lambda}_1$。

(2)需要指明是怎样从表 1 中得到(T_i,X_i)的值的。表 2 中：前 8 列说明了占用时间的划分，这些数据与 R2 有关；行相当于独立的服务器组，这些服务器是在相对应的月初时部署的；3~8 列说明每组服务器每月总的占用时间，是基于第 2 列说明每组服务器的数量数据的；第 2 列数值是表 1 中第 3 列的值连续增长得到的。

表 2 R2 占用时间的分配，服务器组的硬件失效

运行软件 R2 的服务器组	服务器编号	月份及其天数 1 月 31 天	2 月 30 天	3 月 31 天	4 月 31 天	5 月 30 天	6 月 31 天	总占用时间 T_i	分配硬件失效的编号 e_i
1	108	961	930	961	961	930	961	5 740	17
2	145		3 240	3 348	3 348	3 240	3 348	16 524	48
3	99			4 495	4 495	4 350	4 495	17 835	52
4	54				3 069	2 970	3 069	9 108	27
5	21					1 620	1 674	3 294	10
6	458						651	651	2
总计		961	4 170	8 804	11 873	13 110	14 198	53 116	256

表 2 中的第 9 列代表每个服务器组的占用时间，其值就是需要的值；第 10 列是每个服务器组的硬件失效对应的导数，这些数据是由硬件失效总数按比例分配得到的，比率是通过将总服务器组占用时间除以 71 155 个系统日期的总失效时间得到的。

由表 2 可知，在 6 个月期间，运行软件 R2 的服务器有 156 个硬件失效。一些已知的硬件失效实际上是硬软件交互失效。下面，用 F 表示那些实际上是硬软件交互的失效在硬件失效中所占的比例，令 $X_i=F_{e_i}$，其中 e_i 是表 2 中第 10 列中的数值。由于 F 的值是未知的，我们将检验对这个数值分析的精度。对(λ_1,λ_3)的最小二乘估计$(\hat{\lambda}_1,\hat{\lambda}_3)$由表 3 中第 3 和第 4 列给出，这是基于不同的(F,G)值得到的。

表 3 为(F,G)的变值做最小二乘估计$(\hat{\lambda}_1,\hat{\lambda}_3)$

F	G	$\hat{\lambda}_1$	$\hat{\lambda}_3$
0.05	5	1.75×10^{-4}	8.77×10^{-3}
0.10	5	3.51×10^{-4}	1.75×10^{-3}
0.15	5	5.26×10^{-4}	2.63×10^{-3}
0.20	5	7.01×10^{-4}	3.51×10^{-3}
0.25	5	8.77×10^{-4}	4.38×10^{-3}
0.05	10	1.61×10^{-4}	1.61×10^{-3}
0.10	10	3.21×10^{-4}	3.21×10^{-3}
0.15	10	4.82×10^{-4}	4.82×10^{-3}
0.20	10	6.43×10^{-4}	6.43×10^{-3}
0.25	10	8.04×10^{-4}	8.04×10^{-3}

对于式(3)，还要估计纯硬件模型参数和纯软件模型参数。一个很适合 R2 软件失效数据的纯软件失效模型是 G-O 模型，但 G-O 模型的调试不完美。该模型的均值函数 $m(t)=(a/p)(1-e^{-bpt})$，其中 a 是软件初始失效数的期望值，b 表示每个故障的平均失效率，p 是检测到的故障被成功移除的概率。估计参数 $\hat{a}=18.0$，$\hat{b}=6\times10^{-5}$，$\hat{p}=0.95$。

对于纯硬件失效模型，假设每个服务器对应一个韦伯过程。韦伯过程是一种失效率 $h(t)=\lambda\beta t^{\beta-1}$，相应的均值函数 $m(t)=\lambda t^{\beta}$ 的 NHPP 模型。服务器的部署次数不同会导致不同的失效率，令 s_i 表示第 i 个服务器的部署次数，就可以求得均值函数。则第 i 个服务器的失效率和均值函数可以写成：

$$h_i(t)=\lambda\beta(t-s_i)^{\beta-1}I(t\geq s_i)$$

$$m_i(t)=\lambda(t-s_i)^{\beta}I(t\geq s_i)$$

其中 I(.)表示指示函数。

当配备纯硬件模型时，由于硬件对于 R1 和 R2 这两个软件是相似的，故可以使用来自所有服务器的失效数据。在这个例子中，有 6 个不同的起始时间，$s_1\equiv0$，s_2，…，s_6。表 5 的第 2 列给出在每个起始时间运行的服务器数量 n_i，来自于表 2 最后两列的数据。

在不同的观察点 t_i 得到总的失效数据，在观察点，数据是由所有正在运行的服务器集成的。观察点和启动时间的关系如下：$t_1=s_1$，$t_2=s_2$，…$t_{l-1}=s_l$，$t_l=s_l+\triangle$，其中 $\triangle=31$ 天，是新硬件设备最后一次激活到取样末期的持续时间。在每一个观测点 t_i，累计失效数 Q_i 被记录下来，这些值可见表 4 的第 4 列。用这些数据对(λ,β)进行最大可能性估计是不完全的，取而代之的是非线性最小二乘法，这种方法被证明非常有效。

表 4 (λ,β)估计的总体数据

起始时间 s_i	启动的服务器数 n_i	观察时间 t_i	累计失效数 Q_i
0	318	31	23
31	25	61	55
61	20	92	87
92	37	123	108
123	39	153	152
153	19	184	207

定义 $\mu(t)=\sum_{i=1}^{458}\mu_i(t)$，表示经过时间 t 失效数的期望值。由 $t\in\{t_1,\cdots t_6\}$，以及关系式：$t_1=s_2,t_2=s_3,\cdots t_5=s_6,t_6=s_6+\triangle$，可以得到：$\mu(t)=\sum_{j\le i}n_j\lambda(t_i-s_j)^\beta$。在硬软件交互失效模型中，通过最小化二乘错误总数 $SSE(\lambda,\beta)=\sum_{i=1}^{6}[\mu(t_i)-Q_i(1-F)]^2$ 来得到估计$(\hat{\lambda},\hat{\beta})$。表 5 给出了在不同的 F 值下对应的$(\hat{\lambda},\hat{\beta})$($F$ 是在纯硬件失效模型中要被移除的硬件故障数的比例值)。

表 5 对变值 F 做最小二乘估计$(\hat{\lambda},\hat{\beta})$

F	$\hat{\lambda}$	$\hat{\beta}$
0.05	1.25×10^{-3}	1.15
0.10	1.18×10^{-3}	1.15
0.15	1.12×10^{-3}	1.15
0.20	1.05×10^{-3}	1.15
0.25	9.86×10^{-3}	1.15

从表 5 可以得到两个结论：(1)F 的选择对$\hat{\beta}$的影响不大；(2)种种数据表明 β 的实际值要比 1 大。表 6 对 F=0.15 时观测和预测累计失效数进行了比较，可知最小二乘估计比较好。

表 6 F=0.15 时最小二乘估计的吻合度

t_i	$Q_i(1-F)$	$\hat{\mu}(t_i)$
31	19.55	18.45
61	46.75	41.58
92	73.95	68.77
123	91.80	99.79
153	129.20	133.66
184	175.95	171.31

为简化式(3)的运用，(F,G)选中一个值，表 3 就可以确定$(\hat{\lambda}_1,\hat{\lambda}_3)$，这便于估计 $R_{HW}(t)$及 $R_{SW}(t)$；表 5 中的数据可以确定$(\hat{\lambda},\hat{\beta})$，以便求出 $R_{HW}(t)$。对 $R_{SW}(t)$的估计要建立 G-O 软件可靠性模型。

2.3 模型比较

下面将硬件、软件和硬软件交互模型综合起来得到系统的可靠性。假设一个新的服务器在 T=53 086 天时被配备，可以将式(1)中的混合系统可靠性重写为：

相比之下，不考虑硬软件交互失效，系统的可靠性公式为：

$$R_{\text{indenpendence}}(t|T)=R_{HW}(t)\cdot R_{SW}(t)=e^{-\lambda t\beta}e^{-(a/p)}e^{-bpt)}(1-e^{-bpt}) \quad (6)$$

式中，估算 $R_{HW}(t)$的时候，表 5 中的数据反映硬软件失效不是被视为纯硬件失效就是纯软件失效而被忽略掉的影响。

式(5)和(6)联立可得到每一个假定模型下对应的系统的 MTTF 值。表 7 对选择不同(F,G)值的两种模型的 MTTF 的对比，再次表明独立模型比较好。

表 7 系统 MTTF 的对比

F	独立模型	混合模型	
		G=5	G=10
0.05	307.2	305.0	303.9
0.10	321.2	313.0	310.0
0.15	334.1	317.1	312.2
0.20	350.6	321.9	315.3
0.25	366.9	324.4	316.6

许多现有的可靠性模型都没有考虑硬软件子系统的交互失效，而只考虑了纯硬件和纯软件失效。结果，大多数失效数据都是就纯硬件和纯软件失效而得到的。当建立纯硬件或纯软件模型时，硬软件交互失效被忽略。这对于可靠性预测方法是非常不利的。本文方法由于考虑了硬软件交互失效对整个系统的影响，这样建立的模型比传统的只分别考虑软件和硬件所建立的模型具有更准确的可靠性预测。

参考文献

[1] BAIN L J. Statistical analysis of reliability and life-testing models[M]. New York: Marcel Dekker, 1978.

[2] FRIEDMAN M A, TRAN P. Reliability techniques for combined hardware/software systems [C]. Proceedings of Annual Reliability and Maintainability Symposium, 1992: 290-293.

[3] GOEL A L, OKUMOTO K. A Markovian model for reliability and other performance measures of software systems[C]. Proceedings of the National Computer Conference, 1979: 769-774.

[4] HECHT H, HECHT M. Software reliability in the system context [J]. IEEE Transactions on Software Engineering, January, 1986, 12(1):51-58.

[5] IYER R K. Hardware-related software errors: measurement and analysis [J]. IEEE Transactions on Software Engineering, February, 1985,11(2):223-230.

[6] PARZEN E. Stochastic processes [M]. San Francisco, CA: Holden-Day, 1962.

[7] PHAM H, Software reliability [M]. New York: Springer, 2000.

[8] WELKE S R, JOHNSON B W, AYLOR J H. Reliability modeling ofhardware/software systems [J]. IEEE Transactions on Reliability, September, 1995, 44(3):413-418.

(收稿日期:2010-11-15)

作者简介:

胡慧芳,女,1984 年生,硕士,主要研究方向:通信系统的可靠性技术。

沈元隆,男,1947 年生,硕士生导师,主要研究方向:通信系统的可靠性技术。