基于 Nelson 模型的软件安全性评估准则*)

Software Safety Assessment Criterions with the Nelson Software Reliability Model

虞 翊 吴芳美

(同济大学电子与信息工程学院 上海200331)

Abstract Software reliability models generally only consider the occurrence frequency and the occurrence time of failure, and do not differentiate the difference of consequence seriousness caused by software failure. However, in the research field of software safety, not only the frequency, but also the consequence seriousness caused by software failure needs to take into account. In this paper, a new software safety concept is put forward, which is based on the software reliability model and the software equivalent risk. Finally, the software safety with the Nelson software reliability model is discussed and the corresponding safety assessment criterions are defined.

Keywords Software, Reliability, Risk equivalence, Safety, Assessment criterion

软件可靠性模型一般只考虑失效出现的频度和时机,而不区分其可能产生的后果差异。在软件安全性研究领域,人们不但要考虑失效出现的频度,还要考虑失效可能产生的后果严重性。如果我们能寻找出一种方法,它能在相同风险概念的前提下,将软件失效后果的严重性差异转嫁到软件失效出现的频度上,使软件失效具有相同的后果严重性,则在将频度转换成概率后,可用已有的软件可靠性模型来讨论软件的安全性评估准则。本文以 Nelson 模型为例讨论在风险等效前提下的软件安全性概念及相应的评估准则。

1 软件风险度量及软件安全性

定义1 设 $E=\{e_k \mid k=1,2,\cdots,n\}$ 为因软件故障而引发的软件错误输出集,$D=\{d_j \mid j=1,2,\cdots,r\}$ 为因软件错误输出而引发的系统事故集,$P_{ek}=P(e_k)=\sum_{f_i\in F}P(f_i)\times P(e_k/f_i)$ 为软件错误输出 e_k 的概率,F 为软件故障集,$P_{dj/ek}=P(d_j/e_k)$ 为因软件错误输出 e_k 而引发系统事故 d_j 的转移概率,C_{dj} 为事故 d_j 所产生的危害后果,则软件的风险度量可定义为:

$$R_s=\sum_k[P_{ek}\times\sum_j(P_{dj/ek}\times C_{dj})] \qquad (1)$$

B. Malcolm 认为安全性是一个不只局限于时间和空间的概念,对它的态度是主观的并随时间和环境而变化的,对于安全性表述应基于如危害、风险等这些比较客观地定义了的术语[1]。为此,本文使用一种建立在风险度量基础上的软件安全性概念。

定义2 设 R 是某软件的风险度量值,r 是该软件可接受的风险水平,当 $R\leqslant r$ 时,称该软件为软件安全性合格产品,否则为软件安全性不合格产品。

定义2表达了软件安全性是反应软件风险下大于其可接受水平的一种能力。

2 软件风险等效

式(1)可改写为:

$$R_s=\sum_k[P_{ek}\times\sum_j(P_{dj/ek}\times C_{dj}/C)]\times C$$
$$=\sum_k P_k\times C \qquad (2)$$

式中,$R_k=P_{ek}\times\sum_j(P_{dj/ek}\times C_{dj}/C)$ 称为软件第 k 个失效的风险等效出现频度;C 是反映失效后果严重性的一个参数,称失效后果严重性参数。

式(2)是式(1)的一种变形,它们具有相同的计算值,但式(2)将失效后果的严重性差异转嫁到了失效出现的频度上,使系统事故集 $D=\{d_j \mid j=1,2,3,\cdots\}$ 中的元素具有相同的严重性后果,软件失效导致的系统事故概率($P_{dj/ek}$)均为1,所以称式(2)是式(1)的风险等效计算公式。式(1)所表示的系统为实际系统,式(2)所表示的为软件风险等效系统。

软件风险等效系统只是将软件失效后果的严重性差异转嫁到了失效的频度上,它并没有改变软件失效自身的物理意义和软件失效的风险。

在软件风险等效系统中,软件第 k 个失效的出现

*)基金项目:铁道部科技研究开发计划项目(98x12)。虞 翊 副教授、博士。

频度 P_k 为：

$$P_k = P_{ek} \times P_{dj/ek} \times C_{dj}/C + P_{ek} \times P_{d2/ek} \times C_{d2}/C + P_{ek} \times P_{d3/ek} \times C_{d3}/C + \cdots$$

其中第 j 项为：

$$P_{kj} = P_{ek} \times P_{dj/ek} \times C_{dj}/C$$

$P_{ek} \times P_{dj/ek}$ 为经由软件失效 e_k 的实际系统事故 d_j 的出现概率；P_{kj} 为经由软件失效 e_k 的软件风险等效系统事故 d'_j 的出现频度。

如果选取 $C > C_{dj}$，则 $P_{kj} < P_{ek} \times P_{dj/ek}$。其物理意义是：由于选取的严重性参数 C 比实际系统事故 d_j 的严重性参数 C_{dj} 大，所以在放大了实际系统事故 d_j 的严重性参数后，在软件风险等效的条件下，相应的事故出现频度要小一些。

同理，如果选取 $C < C_{dj}$，则 $P_{kj} > P_{ek} \times P_{dj/ek}$，其物理意义是由于选取的严重性参数 C 比实际系统事故 d_j 的严重性参数 C_{dj} 小，因此在缩小了实际系统事故 d_j 的严重性参数后，在软件风险等效的条件下，相应的事故出现频度要大一些。

设：$O_f = \{e_k \mid K = 1, 2, \cdots, K\}$ 为软件的失效集，P_{ek} 为软件失效 e_k 的实际出现概率，则 $P_{EK} = \{P_{ek} \mid k = 1, 2, \cdots, K\}$ 为对应于 O_f 的实际出现概率集。

根据式(2)，$P_k = f(P_{ek})$，所以 P_k 与 P_{ek} 之间是一种映射关系，而且这种映射关系是建立在软件风险等效基础上的，设 $P'_{EK} = \{P_k \mid k = 1, 2, \cdots, K\}$，则 P'_{EK} 是 P_{EK} 的一个建立在软件风险等效基础上的映射集，称为风险等效出现频度集。O_f、P_{EK} 及 P'_{EK} 之间的关系如图1所示，

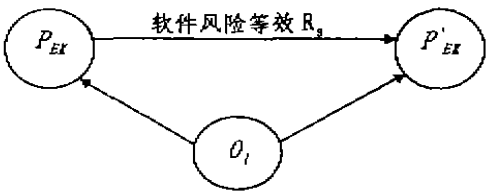

图1　O_f、P_{EK} 及 P'_{EK} 之间的关系图。

设：$O_r = \{e'_l \mid l = 1, 2, \cdots, L\}$ 为软件的正常输出集，P_{el} 为软件正常输出 e'_l 的实际出现概率，则 $P_{EL} = \{P_{el} \mid l = 1, 2, \cdots, L\}$ 为对应于 O_r 的实际出现概率集，且有：

$$\begin{cases} O = O_r \cup O_f = \text{软件输出全集} \\ \sum_{k=1}^{K} P_{ek} + \sum_{l=1}^{L} P_{el} = 1 \end{cases}$$

设：

$$C = \max(C_{d1}, C_{d2}, C_{d3} \cdots)$$

即选取 C 为最大的系统事故后果严重性参数，有：

$$\begin{cases} C_{dj}/C \leqslant 1, j = 1, 2, 3, \cdots \\ \sum_{k=1}^{K} P_k = \sum_{k=1}^{K} [P_{ek} \sum_{j} (P_{dj/ek} \times C_{dj}/C)] < \\ \sum_{k=1}^{K} (P_{ek} \sum_{j} P_{dj/ek}) < 1 \end{cases}$$

此时，可将软件失效的风险等效出现频度 P_k 改称为软件失效的风险等效出现概率，P'_{EK} 为对应于 E_f 的风险等效出现概率集。

在将失效后果的严重性差异转嫁到了失效出现的频度上，即可不考虑软件失效后果的严重性差异之后，又将 P_k 转变成了概率，则此时已具备了用软件可靠性模型来讨论软件安全性问题的条件。

在软件可靠性工程中，有许多模型，如 J-M 模型、S-W 模型、MUSA 执行时间及 Littlewood 贝叶斯排错模型等[2~4]，它们均假设了“程序要在与预期运动环境相似的条件下运行”，而本文提出的软件风险概念是建立在统计基础上的，所以这里选用 Nelson 可靠性统计模型来讨论软件的安全性概念及相应的评估准则等问题。

3　Nelson 可靠性统计模型[5]

Nelson 认为，计算机程序可看成是一个可计算函数 F 的说明，程序的输入数据域 E 为：

$$E = \{E_i \mid i = 1, 2, 3, \cdots, N\}$$

其中 E_i 是使程序运行一次的输入数据，所以 E 是全部输入数据的集合。式中的 N 是集合中的输入数据总数。集合中的每一个元素 E_i 都与程序的一次运行相对应。输入数据 E_i 经程序运行后，得到的输出数据值为 $F'(E_i)$，用 $F(E_i)$ 表示函数的真值，用 Δ_i 表示容许的最大误差，则当条件：

$$|F'(E_i) - F(E_i)| \leqslant \Delta_i$$

得到满足时，程序的运行是成功的；否则程序运行发生失效。

程序的输入数据域 E 可划分为二个子集，E_r 和 E_f：

$$\begin{cases} E = E_r \cup E_f \\ E_r \cap E_f = \Phi \end{cases}$$

式中，E_r 是保证正常运行的输入数据的集合；E_f 是导致运行失效的输入数据的集合。

用 N_f 表示子集合 E_f 中的输入数据的总数，则当程序运行时的数据选用服从均匀分布，且 N 足够大时，程序运行一次的失效概率可定义为：

$$P_f = N_f/N$$

程序成功运行一次的概率为：

$$S = 1 - (N_f/N)$$

程序成功运行 m 次的概率为：

$$S(m)=[1-(N_f/N)]^m$$

当程序运行时的数据选用不服从均匀分布时，可用 P_i 表示 E_i 被选用的概率，这时程序按数据选用概率分布运行一次的失效概率定义为：

$$P_f=\sum_{i=1}^{N}P_iy_i \tag{3}$$

其中

$$y_i=\begin{cases}1 & E_i\in E_f\\ 0 & E_i\in E_r\end{cases}$$

程序成功运行一次的概率为：

$$S=\sum_{i=1}^{N}P_i(1-y_i)$$

如果程序运行 m 次，每次运行输入数据的选择是独立的，则程序不发生失效的概率为：

$$S(m)=[\sum_{i=1}^{N}P_i(1-y_i)]^m$$

当选择的运行输入数据为不独立时，可将输入数据 E_i 在第 q 次运行时被选用的概率表示为 p_{qi}，因此，程序在第 q 次运行中失效的概率为：

$$P_q=\sum_{i=1}^{N}p_{qi}y_i \tag{4}$$

程序在第 q 次运行中成功的概率为：

$$S_q=1-p_q$$

程序运行 m 次不发生失效的概率为：

$$S(m)=\prod_{q=1}^{m}(1-p_q)$$

4 利用 Nelson 模型的软件安全性评估准则

依据软件风险等效概念及 Nelson 可靠性统计模型，式(3)可表示成：

$$P_f=\sum_{i=1}^{N}P_iy_i=\sum_{k=1}^{K}P_k=P_C \tag{5}$$

它代表了程序运行一次时出现失效的风险等效概率。所以程序安全运行一次的概率为：

$$S=1-P_C \tag{6}$$

程序连续安全运行 m 次的概率为：

$$S(m)=(1-P_C)^m$$

如果程序的每次运行不是独立的，则程序在每次运行时的风险将不同，这时用 $R_q=P_{Cq}\times C$ 表示第 q 次运行时的风险。这样，式(4)可表示为：

$$P_q=P_{Cq}=\sum_k P_k(q)$$

$$=\sum_k[P_{ek}(q)\sum_j P_{dj/ek}(q)\times C_{dj}/C)] \tag{7}$$

它代表了程序在第 q 次运行时出现失效的风险等效概率，其中 $P_{ek}(q)$ 为第 q 次运行时软件失效 e_k 的出现概率，$P_{dj/ek}(q)$ 为第 q 次运行时软件失效 e_k 至系统事故 d_j 的转移概率。所以程序在第 q 次运行时处在安全状态的概率为：

$$S_q=1-P_{Cq} \tag{8}$$

程序连续运行 m 次均处在安全状态的概率为：

$$S(m)=\prod_{q=1}^{m}(1-P_{Cq})$$

根据定义2，要求软件的风险(R)≤可接受的风险水平(r)，否则为软件安全性不合格的产品。

在程序运行独立的前提下，式(5)表示了程序运行一次时出现失效的风险等效概率，根据式(2)可得程序运行一次时的风险为：

$$R_1=P_f\times C=P_C\times C$$

设 r 为软件的可接受风险水平，则要求有：

$$R_1=P_C\times C<r$$

$$P_C<r/C$$

由式(6)得：

$$S=1-P_C>1-r/C$$

该式表明了程序安全运行一次的概率要求。

定义3 设 C 为系统事故最大的后果严重性参数，r 为软件的可接受风险水平，且软件的每次运行是独立的，则当软件安全运行一次的概率 S 满足 $S>1-r/C$ 时，称该软件为一次运行安全性合格产品。

当程序连续运行 m 次时，程序出现 $n(n\leqslant m)$ 次失效的概率为：

$$P_{in}=\binom{m}{n}P_C^n(1-P_C)^{m-n}$$

对应的风险为：

$$R_1(n)=P_{in}\times nC=\binom{m}{n}P_C^n(1-P_C)^{m-n}\times nC$$

所以，可得程序连续运行 m 次的总风险为：

$$R_1=\sum_{i=1}^{m}\binom{m}{i}iP_C^i(1-P_C)^{m-i}\times C$$

如要求程序连续运行 m 次的总风险小于可接受风险水平，则需满足条件：

$$\sum_{i=1}^{m}\binom{m}{i}iP_C^i(1-P_C)^{m-i}<r/C$$

定义4 设 C 为系统事故最大的后果严重性参数，r 为软件的可接受风险水平，且软件的每次运行是独立的，则当软件连续安全运行 m 次的概率 $S(m)$ 满足：

$$S(m)=1-\sum_{i=1}^{m}\binom{m}{i}iP_C^i(1-P_C)^{m-i}>1-r/C$$

时，称该软件为 m 次运行安全性合格产品。

在程序运行不独立的前提下，式(6)表示了程序在第 q 次运行时出现失效的风险等效概率，根据式(2)可得程序在第 q 次运行时的风险为：

$$P_{rq}=P_q\times C=P_{Cq}\times C$$

则要求有：

(下转第129页)

CPU 实现(10)。

(6)操作系统(OS)(22核心学时) 内容包括:操作系统原理(2);并发(6);调度和发送(3);虚拟内存(3);设备管理(2);安全和防护(3);文件系统和命名(3);实时系统。

(7)人机接口(HC)(3核心学时) 内容包括:人机接口原理(3);用户建模;交互;窗口管理系统设计;帮助系统;评价技术;计算机支持的协同任务。

(8)图形学、可视化和多媒体(GR)(无核心学时) 内容包括:图形系统;图形技术;基本渲染;基本几何建模;可视化;虚拟现实;计算机动画;高级渲染;高级几何建模;多媒体数据技术;压缩和解压;多媒体应用和内容编辑;多媒体服务和文件系统;网络及分布式多媒体系统。

(9)智能系统(IS)(10核心学时) 内容包括:智能系统的基本论(2);搜索和优化方法(4);知识表示和推理(4);学习;智能代理;计算机视觉;自然语言处理;模式识别;高级机器学习;机器人学;基于知识的系统;神经网络;遗传算法。

(10)信息管理(IM)(10核心学时) 内容包括:数据库系统(2);数据建模和关系模型(8);数据库查询语言;关系数据库设计;事务处理;分布式数据库;高级关系数据库设计;物理数据库设计。

(11)网络中心计算(NC)(15核心学时) 内容包括:网络中心计算介绍(9);互联网(Web)客户/服务器计算的实例(6);建造互联网应用;通信和网络;分布式对象系统;协作技术和群件;分布式操作系统;分布式系统。

(12)软件工程(SE)(30核心学时) 内容包括:软件处理和度量(9);软件需求和规范(6);软件设计和实现(6);验证和确认(6);软件工具和环境(3);软件项目方法学(3)。

(13)计算科学(CN)(无核心学时) 内容包括:数值分析;科学可视化;科学计算体系结构;并行结构程序设计;应用。

(14)社会的、伦理的和职业的论题(SP)(16核心学时) 内容包括:计算历史(1);计算的社会背景(2);分析的方法和工具(2);职业和伦理的职责(2);安全关键系统的风险和责任(2);知识产权(3);隐私和公民权力(2);Internet 的社会影响(2);计算机犯罪;计算中的经济论题;伦理学的哲学基础。

(下转第95页)

(上接第122页)

$R_{eq}=P_{Cq}\times C<r$

$P_{Cq}<r/C$

由式(7)得:

$S_q=1-P_{Cq}>1-r/C$

上述表明了程序在第 q 次安全运行的概率要求。

定义5 设 C 为系统事故最大的后果严重性参数,r 为软件的可接受风险水平,且软件的每次运行是不独立的,则当软件第 q 次运行处在安全状态的概率 S_q 满足 $S_q>1-r/C$ 时,称该软件为第 q 次运行安全性合格产品。

程序连续运行 m 次,在前 $n-1$ 次运行正常、第 n 次运行出现失效的概率为:

$$P_{n/n-1}=P_{Cn}\prod_{k=1}^{n-1}(1-P_{Ck})$$

对应的风险为:

$$R_i(n/n-1)=P_{n/n-1}\times C=P_{Cn}\prod_{k=1}^{n-1}(1-P_{Ck})\times C$$

如要求安全运行 $n-1$ 次后,软件的风险仍处在可接受状态,则要求满足条件

$$P_{Cn}\prod_{k=1}^{n-1}(1-P_{Ck})<r/C$$

定义6 设 C 为系统事故最大的后果严重性参数,r 为软件的可接受风险水平,且软件的每次运行是不独立的,则当软件连续安全运行 $n-1$ 次,在第 n 次仍保持安全运行的概率 $S_{n/n-1}$ 满足:

$$S_{n/n-1}=1-P_{n/n-1}=1-P_{Cn}\prod_{k=1}^{n-1}(1-P_{Ck})>1-r/C$$

时,称该软件为 $(n-1)$ 次安全运行条件下的下一次运行安全性合格产品。

定义3~6分别是对应于一次性独立运行任务、连续 m 次独立运行任务、第 q 次非独立运行任务和在 $n-1$ 次安全运行条件下运行第 n 次任务时的软件安全性评估准则。

参 考 文 献

1 Bob M. Software in Safety-related System: Basic Concepts and Concerns. In: Phil Bennett, ed. Safety Aspects of Computer Control. Butterworth Heinemann Ltd., 1993. 1~18

2 郦萌. 计算机软件的可靠性. 北京:国防工业出版社,1988

3 蔡开元. 软件可靠性工程基础. 北京:清华大学出版社,1995

4 Lyu M R. Handbook of Software Reliability Engineering. Los Alamitos, California: IEEE Computer Society Press, 1996

5 Nelson E. Estimating Software Reliability from Test Data. Microelectronics & Reliability, New York, Pergamon, 1978. 67~74