第34卷 第10期 计 算 机 工 程 2008年5月
Vol.34 No.10 Computer Engineering May 2008
·软件技术与数据库· 文章编号：1000—3428(2008)10—0046—03 文献标识码：A 中图分类号：TP301.6

# 基于加速剖面的软件防危性验证测试方法

覃志东[1]，蔡 勇[2]，王洪亚[1]，刘晓强[1]
（1. 东华大学计算机科学与技术学院，上海 201620；2. 华东计算技术研究所，上海 200233）

**摘 要：** 在严格区分软件防危性和可靠性的基础上，提出一种基于加速剖面的软件防危性验证测试方法。该方法通过系统性的防危分析，构建软件加速剖面，根据重要性取样原理求得测试加速因子，能在减少测试代价的同时实现对软件防危性指标的高可信验证测试。
**关键词：** 软件防危性；防危性验证；软件测试；加速剖面

# Software Safety Demonstration Test Method Based on Accelerating Profile

QIN Zhi-dong[1], CAI Yong[2], WANG Hong-ya[1], LIU Xiao-qiang[1]
（1. School of Computer Science & Technology, Donghua University, Shanghai 201620;
2. Institute of East-China Computer Technology, Shanghai 200233）

**【Abstract】** A software safety demonstration test method based on the accelerating profile is developed on the basis of strictly distinguishing software safety and software reliability. This method includes the details of how to construct the accelerating profile by systematical software safety analysis. According to the method, the software safety target can be demonstrated confidently together with sharply reducing the testing cases size after the accelerating factor is derived by importance sample principle.
**【Key words】** software safety; safety demonstration; software test; accelerating profile

软件防危性(software safety)是指软件作为系统的组成部分，系统不会因为软件的原因而进入危险状态的一种能力。对近年发生的重大灾难性事故的调查研究发现，软件失效是造成这些事故的直接原因[1]。因此，在开发安全关键软件之前都规定了明确的可靠性和防危性验收指标[2-3]，如何高可信地验证这些指标，是学术界和产业界亟待解决的重大难题[4]。

## 1 软件防危性与可靠性的区别

软件防危性、可靠性是可信性的2个子属性，其区别在于：(1)含义不同。可靠性强调的是持续服务能力，是对服务过程质量的一种量化；而防危性强调的是防止危险发生的能力，必要时可停止用户期待的系统功能服务，是对服务结果危害程度的一种量化。(2)关注目标不同。可靠性关注系统的所有失效，特别是发生频率较高的失效；而防危性只关注风险 $Risk$ 较大的失效。如果用 $\varepsilon(hazard)$ 表示事故严重程度，$P(hazard)$ 表示事故频率，则 $Risk=\sum\varepsilon(hazard)\times P(hazard)$，即防危性与失效频率只存在间接关系。(3)不存在等价关系。一个系统可靠不一定能防危，如即将爆炸的核电站继续发电满足其可靠性需求，但与防危性目标相违背；相反，一个系统防危也不一定可靠，如核电系统无论在什么情况下都采取停堆措施，虽具有极高的防危性，但失去了系统应有的可靠性(通常故障下应继续发电)。因此，如图1所示，两者不存在完全包含与被包含的关系。

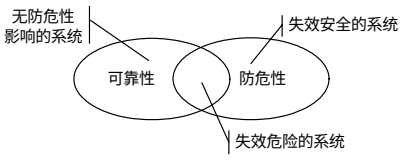


**图1 防危性与可靠性之间关系**

## 2 软件防危性验证测试的困难性

### 2.1 软件防危完善度等级

针对不同的应用场合，对安全关键软件已规定了相应的防危完善度等级(Safety Integrity Level, SIL)，表1为IEC 61508的4级防危完善度标准。

**表1 IEC 61508规定的防危完善度等级**

| 防危完善度等级 | 失效度量 |
|---|---|
| 4 | $10^{-9}$～$10^{-8}$ |
| 3 | $10^{-8}$～$10^{-7}$ |
| 2 | $10^{-7}$～$10^{-6}$ |
| 1 | $10^{-6}$～$10^{-5}$ |

显然，现有的防危性定性评估方法无法对这些SIL进行验证[5]。另外，若采用统计测试方法对防危指标 $(\lambda_0, C)$ 进行验证，无失效测试持续期为

$$t=-\ln(1-C)/\lambda_0 \quad (1)$$

针对不同的防危指标，根据式(1)计算出所需的验证测试持续期，如表2所示，其中的数据单位为h。

**表2 无失效测试持续期**

| $\lambda_0$ | $C$ | | |
|---|---|---|---|
| | 0.90 | 0.99 | 0.999 |
| $10^{-5}$ | $2.3\times10^{5}$ | $4.6\times10^{5}$ | $6.9\times10^{5}$ |
| $10^{-6}$ | $2.3\times10^{6}$ | $4.6\times10^{6}$ | $6.9\times10^{6}$ |
| $10^{-7}$ | $2.3\times10^{7}$ | $4.6\times10^{7}$ | $6.9\times10^{7}$ |
| $10^{-8}$ | $2.3\times10^{8}$ | $4.6\times10^{8}$ | $6.9\times10^{8}$ |
| … | … | … | … |

**基金项目：** 上海市科技攻关计划基金资助项目(06DZ150003)；东华大学青年教师基金资助项目(112-10-0044056)
**作者简介：** 覃志东(1974－)，男，讲师、博士，主研方向：实时计算，可信性计算；蔡 勇，工程师、硕士；王洪亚，讲师、博士；刘晓强，副教授、博士
**收稿日期：** 2008-01-02 **E-mail：** qinzhidong@gmail.com

可见，由于所需测试持续期太长，基于软件运行剖面对软件防危性指标进行验证测试是现实不可行的。

### 2.2 重要性取样原理

设 $X$ 是一随机变量，其概率密度函数是 $f(x)$。$Y=h(x)$ 是随机变量函数，其期望值为

$$\mu=\int_{-\infty}^{+\infty}h(x)f(x)\mathrm{d}x \tag{2}$$

根据 $f(x)$取样产生样本 $\{x_1,x_2,\cdots,x_n\}$ 及 $(y_1,y_2,\cdots,y_n)$，并计算样本均值：

$$\overline{\mu}=\overline{Y}=\frac{1}{n}\sum_{i=1}^{n}y_i=\frac{1}{n}\sum_{i=1}^{n}h(x_i) \tag{3}$$

在安全关键领域中，需要大量样本去获得对$\mu$比较高的统计置信度。但是，根据重要性取样原理，可以增加那些对于估计$\mu$很重要但又很少被选择到的样本$x_i$的采样频率，以减少样本量。

假设重要性较高的样本$x_i$在$g(x)$中被取样的频率较高，用$X'$代表按$g(x)$分布的随机变量，则式(2)被表示为

$$\int_{-\infty}^{+\infty}h(x)f(x)\mathrm{d}x=\int_{-\infty}^{+\infty}h(x)\Lambda(x)g(x)\mathrm{d}x \tag{4}$$

其中，$\Lambda(x)=f(x)/g(x)$ 被称为似然率。

令 $Y'=h(x)\Lambda(x)$，则式(4)可变为

$$\mu=\int_{-\infty}^{+\infty}y'g(x)\mathrm{d}x=E(Y') \tag{5}$$

这样，原先通过对$f(x)$取样估计 $Y$ 的期望变为通过对 $g(x')$ 取样估计 $Y'$的期望。即根据 $g(x')$产生样本集 $\{x_1',x_2',\cdots,x_n'\}$，得到 $(y_1',y_2',\cdots,y_n')$，再计算出

$$\overline{\mu}=\overline{Y'}=\frac{1}{n}\sum_{i=1}^{n}y_i'=\frac{1}{n}\sum_{i=1}^{n}h(x_i')\Lambda(x_i') \tag{6}$$

重要性取样原理为本文的方法提供了统计理论基础。

## 3 软件防危性验证测试方法

软件防危性验证测试是采用测试的方法，通过对测试结果进行分析，得出一个二选一的结论：接受或者拒绝该软件。对于安全关键软件，必须做无失效的防危性验证测试。本文集重要性取样原理和软件防危性分析为一体，提出了基于加速剖面的软件防危性验证测试方法，其步骤如图 2 所示。

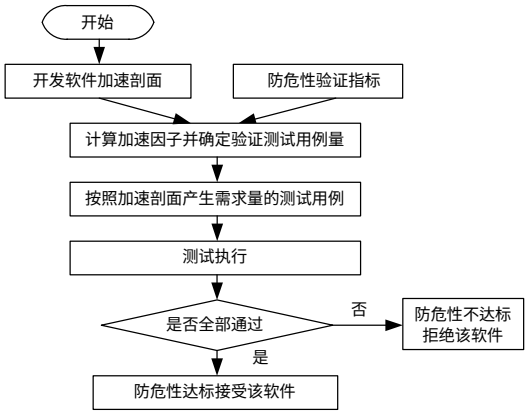


**图 2 软件防危性验证的测试流程**

### 3.1 软件加速剖面的构建

安全关键软件的操作可以分为安全关键集合和普通集合。假设 $oc_1,oc_2,\cdots,oc_n$ 为关键操作，发生的概率分别为 $pc_1,pc_2,\cdots,pc_n$；而 $or_1,or_2,\cdots,or_m$ 为普通操作，发生的概率分别为 $pr_1,pr_2,\cdots,pr_m$，则有

$$\sum_{i=1}^{n}pc_i+\sum_{j=1}^{m}pr_j=P_C+P_R=1 \tag{7}$$

由于 $P_C\ll P_R$，依据运行剖面产生的测试用例主要集中在非安全关键功能上，因此在软件防危性验证测试时，区分关键操作和非关键操作，并依据关键操作产生测试用例，才能有效地压缩测试用例需求量。而软件加速剖面的开发过程就是区分软件关键操作集合和普通操作集合的过程。具体开发步骤如图 3 所示。

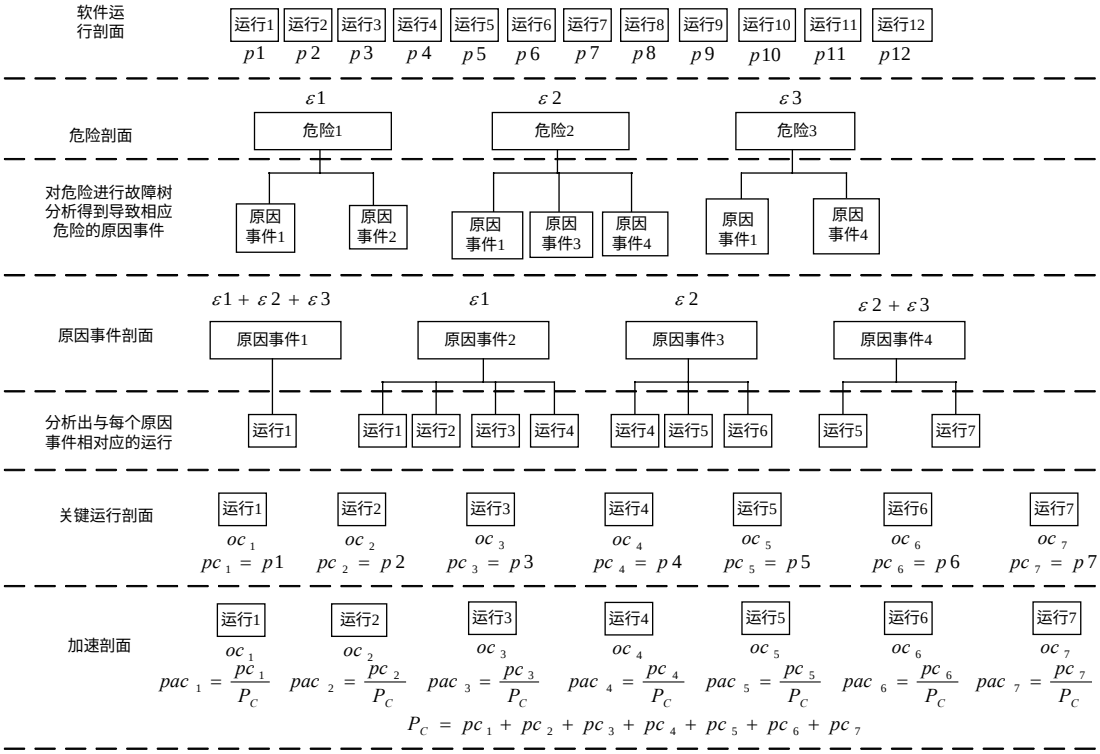


**图 3 软件加速剖面生成步骤**

3.1.1　软件运行剖面的生成

运行剖面代表软件的使用方式。通过运行剖面，可以识别出每个操作发生的概率，进而得到软件的每个安全关键操作在标准状态下的发生概率。

3.1.2　软件危险剖面的生成

软件危险(hazard)是系统处于一种不安全、可能造成事故的状态。能否充分识别出软件危险，并采取相应的措施予以避免，是软件能否防危的基本前提。

软件危险的识别方法有多种，如 PHA 和 HAZOP 等。在危险识别时，需要防危专家、有经验的系统开发专家以及测试人员在一起以头脑风暴讨论会的方式进行，并以检查列表(checklist)的形式尽可能地把软件危险条目一一列举出来。

3.1.3　原因事件剖面的生成

在获得软件危险列表后，以每个危险为顶事件，利用故障树分析法(FTA)对其进行原因事件分析，获得导致这些危险的基本事件。图 4 是一个故障树例子，其割集生成过程如图 5 所示。

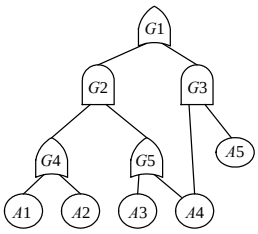


**图 4　故障树模型**

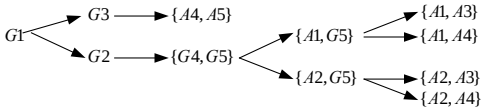


**图 5　故障树割集的生成**

故障树每个最小割集代表一个原因事件。经分析，每个危险都对应一个原因事件集合，不同危险对应的原因事件集合有可能重叠，说明同一事件可能导致多种危险。把得到的所有原因事件集合求并，就得到了软件的原因事件剖面。

3.1.4　安全关键运行剖面的生成

原因事件剖面的每一个原因事件都有与之相关的一个(或多个)软件操作(运行)或操作组合，这些操作一旦激活软件缺陷，就可能触发原因事件，导致危险的发生，所以是安全关键操作$oc_i$。分析每个原因事件都可以得到一个安全关键操作子集合，把所有安全关键操作子集合求并，就可识别出整个软件的安全关键操作集合，而由运行剖面可知其在标准环境下的发生概率$pc_i$，从而生成了软件的安全关键运行剖面。

3.1.5　软件加速剖面的生成

所有安全关键操作发生概率的总和为

$$P_C = \sum_i pc_i \tag{8}$$

则$oc_i$在安全关键集合中发生的相对概率$pac_i$为

$$pac_i = pc_i / P_C \tag{9}$$

关键操作$oc_i$及其对应的相对概率$pac_i$便构成了软件的加速剖面。

### 3.2　基于软件加速剖面的测试用例产生方法

软件防危性验证测试用例的产生步骤如下：(1)将软件加速剖面中所有安全关键运行$oc_i$对应的相对概率$pac_i$求前$j$项和，形成一个数列$\{SC_j\}$，$SC_j = \sum_{i=1}^{j} pac_i$，其中，$j$=1,2,…,$n$，$n$为软件安全关键运行总个数，规定$SC_0 = 0$，并有$SC_1 = pac_1$，$SC_n = 1$，$SC_j - SC_{j-1} = pac_j$。(2)任给一个随机数$\eta \in (0,1.0)$，观察其落在那个区间，若$\eta$满足$SC_{j-1} < \eta \le SC_j$，则该随机数$\eta$与$pac_j$这个概率值对应，而这次随机抽到的安全关键运行为$oc_j$。(3)分析并确定每一个安全关键运行对应的输入变量以及输入变量的取值区间，然后在输入区间随机抽取一个值作为该变量的输入。这样便生成了一个具体的软件防危性验证测试用例。不断地重复以上步骤，直到获得要求的测试用例量。

### 3.3　软件防危性验证测试用例量的决定

从软件加速剖面的开发过程可知，测试加速因子为

$$\Lambda(x) = \frac{\sum pc_i}{\sum pac_i} = P_C \tag{10}$$

假设从软件的输入空间中随机选取一个输入对软件进行一次操作，导致灾难性事故的概率 $pa$ 在软件标准使用状态下的分布可以看作为 Beta 分布：

$$f(pa) = \frac{pa^{a-1}(1-pa)^{b-1}}{B(a,b)} \tag{11}$$

依据软件加速剖面执行$n$个测试用例，发现了$r$个失效。根据式(10)可知，依据软件运行剖面产生测试用例需要$n/P_C$个测试用例才能导致这$r$个失效。那么，$pa$的后验分布为

$$f(pa|r, n/P_C, a, b) = \frac{pa^{a+r-1}(1-pa)^{b+n/P_C-r-1}}{B(a+r, b+n/P_C-r)} \tag{12}$$

对于无先验知识的情况，有$a=b=1$，式(12)变为

$$f(pa|r, n/P_C, 1, 1) = \frac{pa^{r}(1-pa)^{n/P_C-r}}{B(1+r, 1+n/P_C-r)} \tag{13}$$

当从加速剖面产生测试用例时，不容忍灾难事故地验证软件防危性指标$(pa_0, c)$，所需的测试用例量 $N$ 为满足式(14)中 $n$ 的最小整数：

$$P(pa \le pa_0) = \int_0^{pa_0} \frac{(1-pa)^{n/P_C}}{B(1, 1+n/P_C)} \mathrm{d}pa \ge c \tag{14}$$

$$N \approx \left\lceil P_C \frac{\ln(1-c)}{\ln(1-pa_0)} - 1 \right\rceil \tag{15}$$

显然，基于加速剖面需要的测试用例量约为基于软件运行剖面的$P_C$倍，而$P_C$远小于 1。因此，基于软件加速剖面进行软件防危性验证测试能够显著减少测试用例量。

## 4　结束语

本文提出一种集重要性取样原理和防危分析为一体的软件防危性验证测试方法，由软件防危性验证测试的具体实施步骤可知，充分的软件防危性分析是实现高可信的软件防危性验证测试的基本前提，而重要性取样原理为验证测试结果的可信性提供了理论基础。借助形式化的故障树分析方法能有效区分软件的关键运行集合和非关键运行集合、开发软件的加速剖面，切实减少了验证测试的代价，便于工程实践。

## 参考文献

[1] Selding P B. Faulty Software Caused Ariane 5 Failure[J]. Space News, 1996, 7(25): 24-30.

[2] 雍建平, 赵曦滨, 李　晖. 基于软件可靠性工程的测试模型[J]. 计算机工程, 2005, 31(17): 58-61.

[3] 覃志东, 雷　航, 桑　楠, 等. 安全关键软件可靠性验证测试方法研究[J]. 航空学报, 2004, 26(3): 334-339.

[4] DeLong T A, Smith D T, Johnson B W. Dependability Metrics to Assess Safety-critical Systems[J]. IEEE Trans. on Reliability, 2005, 54(3): 498-505.

[5] 徐中伟. 安全软件测试理论与技术的研究及其在铁路信号安全软件测评中的实现[D]. 上海: 同济大学, 1998.