第 35 卷 第 3 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.3 Computer Engineering February 2009
·软件技术与数据库· 文章编号：1000—3428(2009)03—0057—03 文献标识码：A 中图分类号：TP311.5

软件可靠性度量方法

刘晓娟[1]，温冠华[2]，李建军[1]，严少清[1]
(1. 华东计算技术研究所，上海 200233；2. 同济大学计算机系，上海 201804)

摘 要： 分析软件故障暴露率与软件测试次数之间的关系，提出在保证可靠性测试结果客观准确的前提下，有效减少验证测试次数的方法。结合软件可靠性和体系结构相关理论，提出基于组件的软件失效率定量计算方法。研究并利用软件可靠性度量方法，提高了软件可靠性测试效率和可靠性评估准确性。

关键词： 软件可靠性度量；软件可靠性；软件故障暴露率；软件失效率

Method for Software Reliability Measurement

LIU Xiao-juan[1], WEN Guan-hua[2], LI Jian-jun[1], YAN Shao-qing[1]
(1. East China Institute of Computer Technology, Shanghai 200233; 2. Department of Computer, Tongji University, Shanghai 201804)

【Abstract】 This paper analyzes the relationship between the rate of software fault exposure and software testing times. On the premise of objective and accurate results of reliability testing, several methods of how to reduce the total number of testing times are proposed. Based on the theory of software reliability and software architecture, a quantitative computation method of component-based software failure rate is presented. The efficiency of software reliability testing and the accuracy of reliability estimation are improved by the research of software reliability measurement.

【Key words】 software reliability measurement; software reliability; software fault exposure rate; software failure rate

软件系统规模和复杂程度的不断增加使软件可靠性成为影响软件质量的重要因素。对于如何通过软件测试手段对软件可靠性进行客观评估，如何进行基于组件的可靠性测试以及如何提高可靠性测试效率等问题，现有可靠性评估工具在开发技术、适用环境和实用程度上难以满足要求。针对上述问题，本文研究可靠性理论并提出相应的解决方法，为高可信软件的开发、测试与评估提供了新思路。

1 软件故障暴露率与测试次数的关系

定义 1(故障平均检测率) 用规定的方法正确检测到的故障数与故障总数之比，通常用百分数表示[1]。

定理 1 对待测软件 S 进行随机测试，若经过 T 次测试后仍没有故障暴露，则在 T 次测试后，故障暴露的概率 P 满足

$$P=1-(1-K)^T \tag{1}$$

其中，K 为经过 T 次测试后故障的平均检测率；T 为故障暴露前的连续测试次数。

式(1)可以扩展到一般情形，若被测软件 S 中存在 n 个故障 $e_1, e_2, \cdots, e_n$，其平均检测率分别为 $K_1, K_2, \cdots, K_n$，则第 i 个故障 e_i 经过 T_i 次测试后暴露的概率为

$$P_i=1-(1-K_i)^{T_i} \tag{2}$$

故障的平均检测率 K 通常为经验值或根据 PIE 方法计算得到，可以视为定值。下文将根据 P-T 变化规律分析故障暴露概率 P 和测试次数 T 之间的关系。式(1)的 P-T 指数分布曲线如图 1 所示。

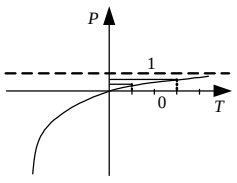

图 1 P-T 指数分布曲线

根据指数函数的性质，由图 1 可以得到如下结论：

(1)随着测试次数 T 的逐渐增加，故障暴露的概率 P 逐渐增大，当测试次数达到一定数值后，P 将趋近于 1。

(2)随着测试次数 T 的不断增加，曲线斜率不断减小，因此，当测试次数 T 的增量 ΔT 相同时，故障暴露概率 P 的增量 ΔP 会逐渐变小。

根据软件测试结果，可以给出软件可靠性，但在软件可靠性测试中，测试结果通常受测试时间的制约。如果可靠性要求较高，就需要大量测试次数，即需要大量测试时间。

本文针对上述问题，结合以上 2 点结论，给出在不降低验证测试结果可信性水平的前提下，有效减少验证测试次数，提高测试效率的 2 种方法：

(1)根据第(1)条结论，给定故障的平均检测率 K 后，根据函数值 P 趋近于 1，测试人员可以判断大约需要几次测试，可以使故障暴露，以便预估测试结束时间，避免过多测试。

由式(1)推导得故障的平均检测率 $K=1-(1-P)^{1/T}$，该值代表 T 次测试后故障被检测到的情形，当故障未被检测出来时，平均检测率 K 必定小于计算值。因此，实际检测率 $K \leqslant 1-(1-P)^{1/T}$，推导得到如下关系式：

$$T \leqslant \log_{1-K} 1-P \tag{3}$$

由式(3)可知，给定故障的平均检测率 K 后，若已知故障暴露概率 P，就可以近似估算最多进行多少次测试能检测出故障。平均检测率可以在需求规格说明书中被定义，也可以是执行前一轮测试后的结果。

例 1 若待测软件 S 中存在 2 个故障 e_1 和 e_2，其平均检测

作者简介： 刘晓娟(1983－)，女，硕士研究生，主研方向：软件测试；温冠华、李建军，硕士研究生；严少清，高级工程师

收稿日期： 2008-10-22 **E-mail：** wenyaooo@hotmail.com

率分别为 K_1=0.96, K_2=0.000 03，则根据式(3)可以得到故障完全暴露(假定 P=0.999 ≈ 1)所需要测试次数为

$$T_1 \leqslant \log_{1-0.96} 1-0.999 \approx 2$$

$$T_2 \leqslant \log_{1-0.000\ 03} 1-0.999 \approx 230\ 255$$

可见，最多经过 2 次测试，故障 e_1 就会暴露；最多需要进行 230 255 次测试，故障 e_2 才会暴露。现实中，在有限时间内，如果已进行上百次测试，多数故障已暴露，就可以停止测试，放弃检测类似 e_2 的故障。这是因为需要经过数十万次测试才会暴露的故障，对于一般的软件开发方和用户而言是可以接受的。

(2)根据第(2)条结论，若给定平均检测率 K，测试人员可以通过逐级增加测试次数 T，分析并比较故障暴露概率 P 的增量 ΔP ，估测故障完全暴露大概需要的测试次数。

例 2 假设待测软件 S 中存在故障 e，其平均检测率 K=0.1，则根据式(1)计算得到 T 次测试后故障暴露的概率 P，具体数据见表 1。

表 1　*P-T* 数据

测试次数 T/次	测试次数增量 ΔT /次	故障暴露概率 P	故障暴露概率增量 ΔP
10	10	$1-(1-0.1)^{10}=0.651\ 00$	0.000 00
20	10	$1-(1-0.1)^{20}=0.878\ 00$	0.227 00
30	10	$1-(1-0.1)^{30}=0.958\ 00$	0.080 00
40	10	$1-(1-0.1)^{40}=0.985\ 00$	0.027 00
50	10	$1-(1-0.1)^{50}=0.995\ 00$	0.010 00
100	50	$1-(1-0.1)^{100}=0.999\ 97$	0.004 97

由表 1 可以看出，当测试次数 T 的增量 ΔT 均为 10 时，故障暴露概率 P 的增量 ΔP 逐渐变小。比如，进行 20 次测试和进行 10 次测试相比，P 增加了 0.227；进行 50 次测试和进行 40 次测试相比，P 增加了 0.01。当测试次数达到一定数量后，故障暴露概率的增长速度会明显降低，进行 100 次测试和进行 50 次测试相比，测试次数的增量 ΔT 为 50，而故障暴露概率仅增加了 0.004 97。根据实际情况，0.995 的暴露概率已接近完全暴露概率，可以认为经过 50 次测试后，若故障未暴露，就没有必要继续测试。

定理 2 对待测软件 S 进行随机测试，假设存在 n 个相互独立的故障 $e_1, e_2,\cdots,e_n$，它们在 $T_i(i=1,2,\cdots,n)$次测试后暴露的概率分别为 $P_1, P_2,\cdots, P_n$，则故障暴露的平均概率 $\overline{P}$ 为

$$\overline{P}=P(\bigcup_{i=1}^{n}e_i)=1-P(\bigcap_{i=1}^{n}\overline{e_i})=1-\prod_{i=1}^{n}(1-P_i) \tag{4}$$

将式(2)代入式(4)，可得

$$\overline{P}=1-\prod_{i=1}^{n}(1-P_i)=1-\prod_{i=1}^{n}(1-K_i)^{T_i} \tag{5}$$

其中，K_i 表示经过 T_i 次测试后故障的平均检测率；T_i 表示故障 e_i 暴露前的连续测试次数。

根据定理 2 可以得到提高测试效率的第 3 种方法：在已知 T_i 次测试后故障平均检测率的前提下，利用式(5)计算故障暴露的平均概率，通过与经验值进行比较，估算被测软件中是否有残存故障，从而判断是否需要终止测试。

例 3 假设对待测软件 S 进行 15 次测试，在测试次数为 T_1=5, T_2=10, T_3=15 时分别检测到故障 e_1, e_2, e_3，其平均检测率分别为 K_1=0.05, K_2=0.04, K_3=0.03，由经验值得到 15 次测试后，软件中故障的平均检测率 K=0.75。若软件中仅存在上述 3 个故障，则根据式(5)可得

$$\overline{P}=1-(1-K_1)^{T_1}(1-K_2)^{T_2}(1-K_3)^{T_3}\approx 0.67<0.75$$

由于故障暴露的平均概率小于平均检测率的经验值，因此软件中仍然有未被检测出的故障，需要继续测试。随着故障暴露个数 n 的增加，$\prod_{i=1}^{n}(1-K_i)$ 的值会减小，故障的平均暴露概率会增大，将趋近于经验值。

2　基于组件的软件失效率

上述方法可以预测一个简单待测软件的失效率，但待测软件通常存在多个模块，且模块间存在各种的联系。要对由若干模块构成的系统进行失效率分析预测，结构分解是关键步骤之一。下文将根据几种典型系统可靠性模型[2-3]对失效率进行研究。

定义 2 失效率[4]是在 t 时刻尚未发生失效的条件下，在 t 时刻后单位时间内发生失效的概率。

2.1　串联系统的失效率

设系统 S 由 n 个子系统(或配置项)组成，如果当且仅当 n 个子系统(或配置项)全部正常工作时，系统才正常工作，任意一个子系统(或配置项)的失效都将导致系统失效，则称 S 是 n 个子系统(或配置项)组成的串联系统，串联系统可靠性模型如图 2 所示。

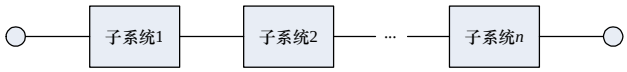

图 2　串联系统可靠性模型

定理 3 对于由 n 个相互独立的子系统组成的串联系统，任何一个子系统(或配置项)失效意味着整个系统失效，因此，在 t 时刻整个系统尚未发生失效的条件下，在 t 时刻后发生失效的概率 $Z(t)$满足以下关系式：

$$Z(t)\cong\max\{Z_i(t)\,|\,i\in[1,n]\} \tag{6}$$

其中，$Z_i(t)$表示第 i 个子系统在 t 时刻后发生失效的概率。

例 4 假设待测软件 S 包含 3 个串联子系统 S_1, S_2, S_3，在 t 时刻整个软件尚未发生失效的条件下，在 t 时刻后单位时间内子系统发生失效的概率分别为 $Z_1(t)$=0.01, $Z_2(t)$ =0.02, $Z_3(t)$=0.03，根据式(6)可以得到整个软件的失效率为

$$Z(t)\cong\max\{Z_1(t),Z_2(t),Z_3(t)\}=0.03$$

由式(1)可知，对系统 S 进行随机测试，若经过 t 时间仍然没有故障暴露，则在 t 时刻后故障暴露的概率 $P(t)$为

$$P(t)=1-(1-K)^{t/T_L} \tag{7}$$

其中，K 为平均检测率；t 为故障暴露前的测试总时间；T_L 为测试的平均执行时间。

根据定义 2 中失效率的概念，结合式(6)和式(7)，得到由 n 个相互独立的子系统组成的串联系统的失效率 $Z(t)$满足以下关系式：

$$Z(t)=P(t)\cong\max\{P_i(t)\,|\,i\in[1,n]\}=\max\{1-(1-K_i)^{t/T_{Li}}\,|\,i\in[1,n]\} \tag{8}$$

因此，只要知道每个子系统的平均检测率 K_i 和平均执行时间 T_{Li}，就可以得到每个子系统在 t 时刻后失效的概率，进而求得整个串联系统在 t 时刻尚未发生失效的条件下，在 t 时刻后发生失效的概率。

2.2　并联系统的失效率

设系统 S 由 n 个子系统(或配置项)组成，如果当且仅当 n 个子系统(或配置项)全部失效时，系统才失效，任意一个子系统(或配置项)正常工作都可保证系统正常工作，则称 S 是 n 个子系统(或配置项)组成的并联系统，并联系统可靠性模型如图 3 所示。

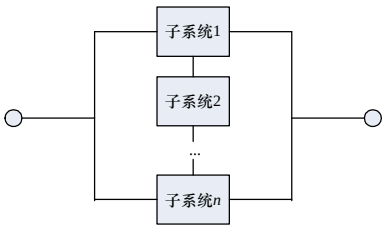

图 3　并联系统可靠性模型

定理 4 对于由 n 个相互独立的子系统组成的并联系统，所有子系统(或配置项)全部失效时，系统才失效，因此，整个系统在 t 时刻尚未发生失效的条件下，在 t 时刻后发生失效的概率 $Z(t)$满足以下关系式：

$$Z(t) \cong Z_1(t) \times Z_2(t) \times \cdots \times Z_n(t) = \prod_{i=1}^{n} Z_i(t)\ ,\ i=1,2,\cdots,n \qquad (9)$$

其中，$Z_i(t)$表示第 i 个子系统在 t 时刻后发生失效的概率。

例 5 假设待测软件 S 包含 3 个并联子系统 S_1, S_2, S_3，在 t 时刻整个软件尚未发生失效的条件下，在 t 时刻后单位时间内子系统发生失效的概率分别为 $Z_1(t)=0.01$，$Z_2(t)=0.02$，$Z_3(t)=0.03$，根据式(9)可以得到整个软件的失效率为

$$Z(t) \cong Z_1(t) \times Z_2(t) \times Z_3(t) = 0.000\ 006$$

与串联系统类似，可以得到由 n 个相互独立的子系统组成的并联系统的失效率 $Z(t)$为

$$Z(t) = P(t) \cong P_1(t) \times P_2(t) \times \cdots \times P_n(t) = \prod_{i=1}^{n} P_i(t) = \prod_{i=1}^{n} 1-(1-K_i)^{t/T_{Li}} \qquad (10)$$

其中，$i=1,2,\cdots,n$。

因此，只要知道每个子系统的平均检测率 K_i 和平均执行时间 T_{Li}，就可以得到每个子系统在 t 时刻后失效的概率，进而求得整个并联系统在 t 时刻尚未发生失效的条件下，在 t 时刻后发生失效的概率。

2.3　混联系统的失效率

模块间的关系通常存在混联关系。混联系统是指在其可靠性依赖关系中同时存在串联和并联 2 种情况的系统，其可靠性模型见图 4。

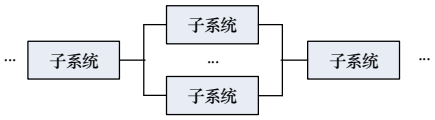

图 4　混联系统可靠性模型

对混联系统失效率的计算可以采用递归方法：先按串联或并联的形式，对系统进行逐级分解，直到分解所得的每个部分均含有单纯串联或并联的形式为止，然后利用上述串(并)联系统失效率计算方法逐级回溯，最终计算得出整个系统的失效率。

3　结束语

本文介绍基于组件的软件失效率计算方法，提出提高软件可靠性测试效率和可靠性评估准确性的可行办法，为软件可靠性度量和模块化测试提供了理论指导。下一步的研究内容包括以下 2 点：(1)引入故障等级概念；(2)引入故障排除率概念。上述 2 个概念的引入可以增强分析结果的实际意义。

参考文献

[1] Musa J D. Operational Profiles in Software Reliability Engineering[J]. IEEE Transactions on Software Engineering, 1993, 10(2): 14-32.

[2] Krishnamurthy S, Mathur A P. On the Estimation of Reliability of A Software System Using Reliabilities of Its Components[C]//Proc. of the 8th International Symp. on Software Reliability Engineering. Albuquerque, USA: IEEE Computer Society, 1997.

[3] 郭　波，武小悦. 系统可靠性分析[M]. 北京：国防科技大学出版社, 2002.

[4] 徐仁佐. 软件可靠性工程[M]. 北京：清华大学出版社, 2007.

（上接第 47 页）

间关系等语法信息，更符合潜在语义聚类的标准；(2)相比由单个词形成的侧面名，由词对形成的侧面名具有更大的信息量，更容易体现侧面内容，更易得到认可。例如，在“洛克比空难”这一语料中，词特征模型得到的新侧面有“政府”、“事件”、“泛美”等，而词对特征模型得到的相对应侧面为“包括_政府”、“制造_事件”、“泛美_破产”等，词对更表现出侧面的特征，不易产生歧义。

7　结束语

本文以基于事件框架信息抽取的研究成果为基础，阐述了引入事件新侧面探测的必要性，为充分利用新闻段落中重要的语法信息，实现了词对特征模型和潜在语义分析聚类算法。测试结果证明了其有效性。未来的研究方向是：

(1)提高生成新侧面的准确率与召回率，主要在于特征选择和聚类算法改进以及确保将已有侧面内容在新侧面探测前除去。

(2)通过对类的共有新侧面的发现，构建一种能根据反馈、自我完善的事件类框架。其难点在于侧面关键词的自动产生。

参考文献

[1] 梁　晗，陈群秀，吴平博. 基于事件框架的信息抽取系统[J]. 中文信息学报, 2006, 20(2): 40-46.

[2] 林鸿飞，宋　丹，杨志豪. 基于语义框架的话题跟踪方法[C]//中文信息处理前沿进展-中国中文信息学会二十五周年学术会议论文集. 北京: [出版者不详], 2006.

[3] 李　芳，毛顺福，蒋德良，等. 中文新闻事件要素抽取研究[C]//全国计算机大会论文集. 苏州: [出版者不详], 2007.

[4] Han Ying, Li Fang. Template-based Chinese News Event Summarization[C]//Proc. of the 2nd Int'l Conf. on Semantics, Knowledge and Grid. Washington D. C., USA: IEEE Computer Society, 2006.

[5] 居　斌. 潜在语义标引在中文信息检索中的研究与实现[J]. 计算机工程, 2007, 33(5): 193-196.