统计分析在软件测试中的应用

温鹏程

(武汉精伦电子股份有限公司,湖北 武汉 430074)

摘 要:运用统计分析技术,根据测试得到的数据,对软件测试本身的质量、软件质量、软件过程质量进行量化分析,找出薄弱环节,作为改进和优化软件开发各个阶段的依据。通过对软件的分析,结合实际工作中常用的统计理论和技术,描述了可行性较高的测试统计分析技术。

关键词:测试质量;统计分析;软件过程质量;软件度量

中图法分类号:TP311.56 **文献标识码**:A

1 引 言

在软件测试时,并不是完成了测试报告就算工作完成了。实际上在分析测试数据的基础上,进行统计分析和数据挖掘,找出问题的规律和深层原因,可以为决策提供指导。根据问题分布的模块、性质和解决情况的统计分析,项目管理者可全面了解产品开发的进度、质量和应该重点聚焦的问题。

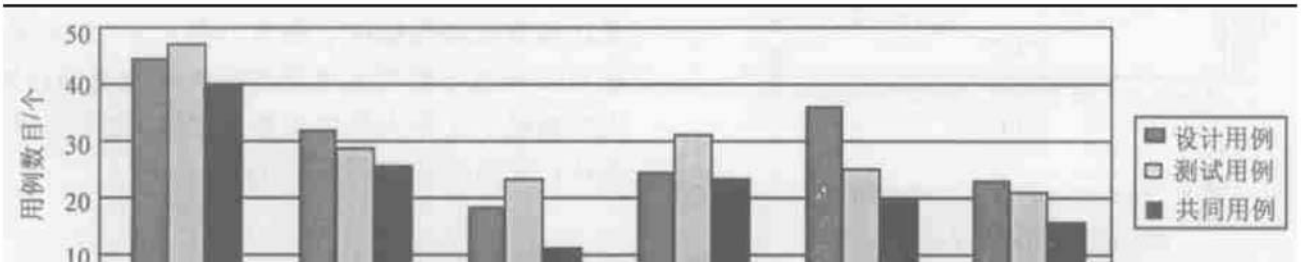
影响质量问题的因素很多,但其中只有少数(20%)因素对质量问题起决定性作用,即关键问题或高危区域[1]。质量正是建立在对测试结果的评估和测试过程分析的基础上的。通过对测试数据的分析,抓住关键问题加以改进,会取得事半功倍的成效,质量问题也会大大减少。

软件测试中的统计分析,做为软件度量的一个分支,有许多成熟的度量模型,如S模型、指数模型等,但因为使用复杂,在实际使用中可操作性不高。利用简单的统计报表和各种简易图形,可以在测试质量分析、软件质量分析、软件过程分析和软件测试管理中进行直观明了的分析。笔者所用数据来自所测商务信息多媒体软件,该软件是基于Linux开发的嵌入式系统。

2 在测试质量分析中的应用

2.1 分析测试用例对设计用例的覆盖度

主要指对设计中的功能和异常处理、并发控制等各种情况的覆盖度,用测试用例占设计用例的比值来衡量,既衡量开发设计又衡量测试设计。如设计用例的数目为M,测试用例的数目为N,共同的数目为L。当$L:M>1$时,说明设计用例考虑不周,意味着需求不明或设计有严重漏洞,如果项目到了评审阶段还有该问题,则说明项目监控和需求评审也未到位。如果这个比例大于1.2,其严重后果甚至可能导致项目终止。当$L:M<1$时,说明测试用例涵盖的范围不够,需要测试设计人员熟悉设计文档,补充测试用例。理想的比值应



作者简介:温鹏程(1972-),男,甘肃天水人,武汉精伦电子股份有限公司工程师.

在 0. 9~ 1. 1 之间。

期望的测试用例是既涵盖设计用例, 又有少部分超出设计用例, 他们的关系应该是异步递增, 互为参考的。图 1 是多媒体信息电话基于功能的设计用例和测试用例在首次评审前的对比分布统计图(部分功能) , 可以看出, 名片簿的需求定义显然不充分, 而文件管理的测试用例远远不够。

2. 2　分析测试用例对场景的覆盖度

场景是一系列用例按照一定的逻辑组成的完整事务, 这种逻辑可以基于实际应用, 也可由限制因素决定, 包含主流程和各种辅助流程、异常处理流程[2]。理解被测软件的功能、运行过程和使用方式后, 用状态图、功能执行表、数据流图、控制流图等清晰规范地表示出来, 对整个软件运行过程的分析和测试就转化为对各个状态下所执行功能的分析和测试, 以利于完成对各个状态和操作流程的完整检测。根据事务对状态、控制、功能的覆盖和场景描述的对比, 检查测试用例对场景的覆盖度, 也可从宏观的角度标明系统复杂度。严格地讲, 所有的事件和控制流都要测试一遍, 但有些功能的状态、事件、控制流的数目十分巨大, 全部测试是不现实的。但关键路径、软件运行时经常执行的路径、失效时可能造成重大损失的路径是必须测试的。表 1 是计算器的状态、事件、控制流、数据

	状态数	事件数	控制流数	数据流数	场景数
设计/ 个	5	14	77	12	10
测试/ 个	5	12	60	12	12
比例/ %	100	86	78	100	120

3　在软件过程质量分析中的应用

软件过程质量决定软件质量。如何在软件生产过程中保证软件过程的质量和效率其实比单纯的产品检验具有更重要的意义。CMMI 模型主张在开发过程中注重对过程和产品的度量, 以量化的形式提供对管理过程的支持, 以及对过程进行相应的评估和改进。这与传统的软件测试的不同之处就在于关注对软件测试结果数据的分析和利用, 以及对整个软件开发过程的关注, 将测试数据转换成为能够标识过程缺陷的统计数据[3]。以测试中各个阶段发现的 bug 数据为基础进行分析, 根据不同的关注侧面, 对软件过程进行度量, 剖析软件过程中存在的问题和出现问题的原因。通过汲取教训, 总结经验, 寻找此类项目的最优的过程组织管理方式, 作为后续项目的参考依据甚至模式, 可以少走许多弯路, 从而可以有效缩短开发周期, 降低质量风险, 提供过程改进的支持。

(1) 分析各个阶段各个模块发现错误的总体走势和关键问题。通过对历史 bug 数据的分析, 找到软件的高危区、中危区、低危区和稳定区。对不同区采用不同的测试策略, 如高危区涵盖较多的关键问题, 投入更多的资源来重点测试; 中危区和低危区投入不同比例的资源进行测试, 面对稳

化。主要的原因是对该功能子域修改时波及面过大, 软件结构化程度不高, 扩散引起新问题, 该功能子域要在下一轮测试中进行重点测试。总之, 通过对软件问题分布进行分析并找出问题分布的高危区, 能够帮助制定测试策略, 指导测试过程。

表 2　软件小批量生产前发现的 bug 的阶段分布情况

模块	需求分析	设计	开发	总计
电话功能	[11, 3, 9, 10]	[15, 5, 8, 30]	[96, 44, 80, 42]	[122, 52, 97, 82]
浏览器	[5, 1, 0, 0]	[1, 1, 1, 15]	[21, 7, 15, 21]	[27, 9, 16, 36]
电子邮件	[23, 15, 16, 30]	[35, 28, 20, 48]	[107, 30, 58, 45]	[165, 73, 94, 123]
名片簿	[8, 3, 5, 20]	[13, 5, 10, 18]	[26, 12, 23, 45]	[47, 20, 38, 83]
日程表	[5, 2, 1, 13]	[8, 3, 5, 21]	[11, 6, 8, 40]	[24, 11, 14, 84]
记事本	[3, 0, 3, 5]	[7, 1, 7, 15]	[35, 19, 21, 37]	[45, 20, 41, 57]
文件管理	[4, 1, 3, 20]	[7, 3, 3, 6]	[5, 3, 0, 0]	[16, 7, 6, 26]
设置	[3, 1, 3, 10]	[11, 3, 8, 22]	[7, 2, 7, 32]	[21, 6, 18, 64]
总计	[62, 26, 40, 108]	[97, 49, 62, 175]	[308, 123, 212, 262]	[467, 198, 314, 545]
比例/ %	[13. 3, 13, 12. 7, 20]	[20. 8, 24. 7, 19. 7, 32]	[65. 9, 62. 3, 67. 6, 48]	—

表 2 是该软件小批量生产前发现的 bug 的阶段分布情况。数据以四维坐标表示,(已发现错误数, 严重错误数, 已修改错误数, 各模块修改错误工作量(人天))。根据表 2 评估哪个阶段的工作不够细致, 以及带来的损失和残余缺陷的影响。根据经验和历史数据, 评估下一阶段的工作量。

从表 2 看, 需求和设计阶段的错误所占比重都超出了合理的比值, 尤其是需求分析和设计阶段造成的问题, 修改工作量比开发阶段解决问题的工作量还大, 这不仅造成反复修改, 也使软件在多次修改后植入更多的潜在 bug。同时可以判断, 该软件稳定运行的较为准确的工作量, 但因可能引入新的错误, 该日期要扩大到 1. 2 倍。WN = 〔所有错误数- 所有已经修改错误数〕× 平均工作量, 修改目前的问题仍然需要(467- 314) × (545/ 314) = 266 人天。通过统计, 可以分别了解由需求定义不明、修改扩散、设计不合理引起的错误数目和带来的资源损失, 定期执行评价, 作为软件测试规划和停止测试的依据, 也作为判断软件项目是否应当停止和改进的依据。

通常情况下, 随着测试接近尾声, 各功能子域中发现的问题应越来越少。如果在整个测试周期的后期, 某些功能子域中仍然发现较多严重问题, 则说明要么这些功能子域存在的问题比较大, 需要重点测试以保证整个系统的质量; 要么该测试周期的时间安排不合理, 需要修改测试计划, 延长测试时间。目前绝大多数软件按照螺旋形进行发布和管理, 为了保证整个软件的质量, 同时确保按计划发布该软件, 会把问题很多的功能子域从整个软件产品中去掉而留在下一版本再实现[4]。

可以将缺陷数按测试历史时间列出, 创建缺陷趋势图; 也可以将缺陷数作为缺陷密度报告中的严重性或状态参数, 分别为揭示软件可靠性的缺陷趋势或缺陷分布提供依据。预期缺陷发现率将随着测试和修复进度而减少, 在某模块缺陷率低于设定阀值时才部署该模块到软件中。根据各功能模块在各个时间段的错误走势, 可以分析不同时期不同人员负责模块的错误分布状态。图 2 是聊天模块在 6 轮测试中各级别的错误走势。致命错误的优先级最高, 依次递减。

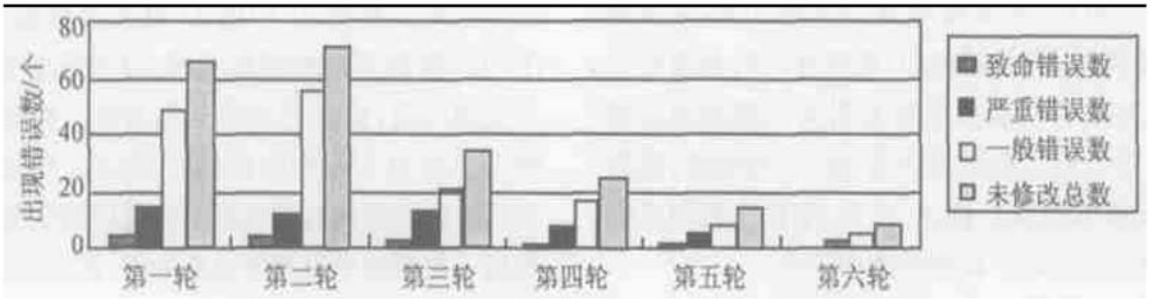


图 2 聊天模块测试周期内错误分布图

(2) 度量版本控制的有效性。实际工作中, 常因为开发历时太长、人员众多、文件数量庞杂, 导致错误发布的各个模块的版本搭配不一致, 测试反复数次或已经修改的问题总是反复出现[5]。虽然使用了各种先进的版本管理工具, 仍然不能彻底杜绝问题, 对这种现象进行统计分析, 找出错误提交的模块, 督促相应人员加以改进, 设立一定的检入保障确认机制。例如在图 3 中, 7 月份以前, 版本发布一直存在问题, 后设立了版本控制专门人员并进行了检入确认, 问题基本杜绝了。

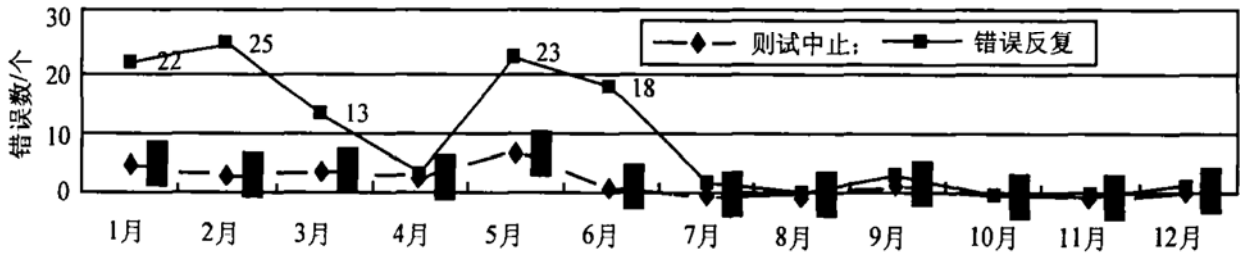


图 3 每月测试中止和错误反复数目

试修改了的模块, 肯定会遗漏一部分问题, 理想的策

析设计文档中各个模块的控制和数据流的交互以及

模块配置图, 能够分析出理论上存在的关联。而根据历史 bug 数据, 运用散布图对各个模块之间的相关性进行分析, 可以挖掘深层相关性。图 4 是修改电话模块 m 处后, 与记事本模块在 2 个测试周期内的相关性图。可以看出, 记事本和电话模块弱相关, 因此, 电话修改后, 记事本需要进行验证性测试。

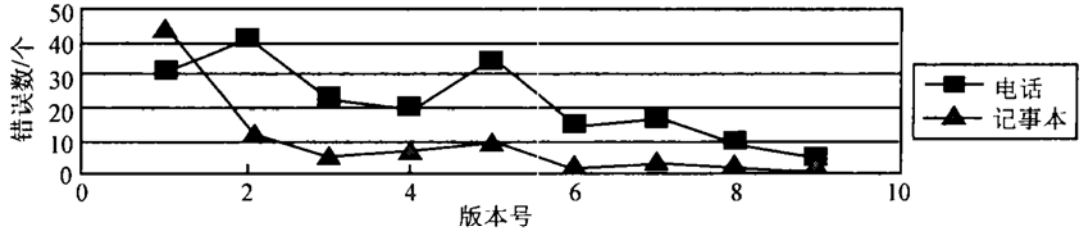


图 4　电话和记事本相关性

4　评估使用环境对软件的影响

实际使用环境很复杂, 难以在测试中对所有环境或环境组合进行模拟, 但必须确定影响系统运行的主要因素和可能出现的意外情况, 在软件中对这些因素主动加以回避, 提高软件的容错能力和自恢复能力。嵌入式软件的运行对环境因素尤其敏感。为了逐一确定各个因素的影响, 先去掉其他影响因素, 只选考察因素, 然后与不加该因素时的软件, 用同一版本进行对比测试, 待确定主要影响因素后, 将所有的主要影响因素都同时进行考察, 做为最极端的环境。同时分析可能互相影响的因素, 进行组合测试和干涉性评估。图 5 是电话功能对外界环境影响的分析图。从图 5 可以看到, 加衰减线并加反极线路时, 出现的错误与不加以上因素时, 显然不同, 需要重点考察。

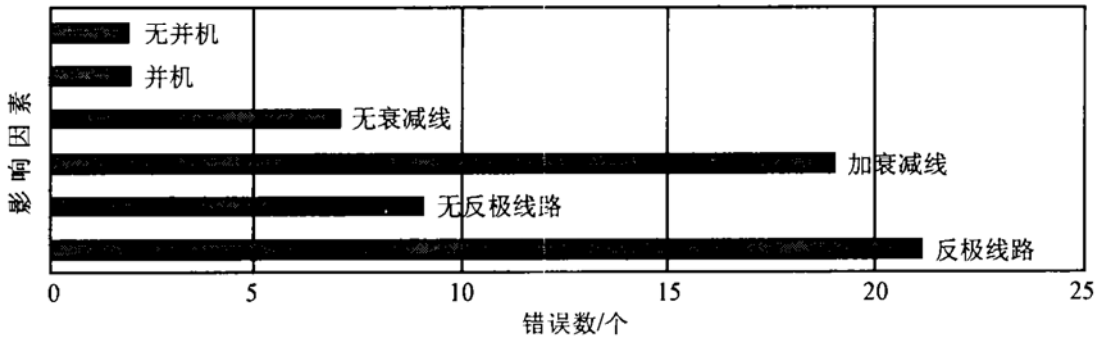


图 5　电话模块影响因素分析图

5　结束语

对同一来源的测试数据, 以不同的视角进行分析, 得到不同的关注面, 从而分析不同的因素的性状。它既可反映软件质量, 也可反应软件过程质量。以具体的量化指标来改进各个标识需要改进的环节, 可以提高软件项目组的整体质量。

参考文献:

[1] Stephen H K. Metrics and Models in Software Quanlity Engineering[M]. 北京: 机械工业出版社, 2003.

[2] Norman E F. 软件质量工程的度量与模型[M]. 北京: 机械工业出版社, 2003.

[3] McGarry J. Pratical Software Measurement[M]. 北京: 机械工业出版社, 2003.

[4] 徐　红, 党月胜, 车向东. 统计过程控制方法在软件测试过程分析中的应用[J]. 计算机工程与应用, 2001, 37(12): 96- 100.

[5] 钱红兵, 刘　超, 晏海华, 何智涛. 基于量化分析的软件测试过程的控制技术[J]. 北京航空航天大学学报, 2001, 27(4): 461- 464.

[6] 殷海风, 史　立. 基于树的分析技术在软件测试中的应用[J]. 微机发展, 1998(3): 44- 46.

Application of Statistics in Software Tests

Wen Pengcheng

Abstract: Using statistics and analysis technique, the quality of the test, the quality of the software and the quality of the software process can be qualitified through the test data. The weakness can be found to be the gist to improve the development in every stage. By information telephone software and the commonly used statistics theory and technique, the test statistics analysis technique with high feasibility is described.

Key words: test quality; statistics analysis; software process quality; software measurement

Wen Pengcheng: Engineer; Wuhan Jinglun Electronic Co., Ltd, Wuhan 430074, China.

[编辑: 王志全]