

来自硅谷软件公司的一线软件测试者的心得之作

实用软件测试

——来自硅谷的技术、经验、心得和实例

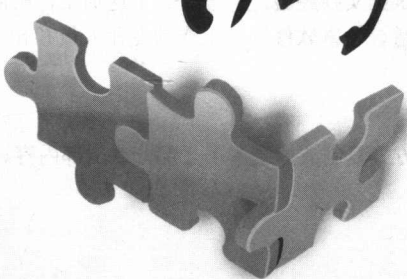
李幸超 著



电子工业出版社

Publishing House of Electronics Industry
<http://www.phei.com.cn>

实用软件 测试



—— 来自硅谷的技术、经验、心得和实例

李幸超

江苏工业学院图书馆
藏书章

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书系统地介绍了有关软件测试的基本知识和全过程。作者从软件测试的规划和方案设计,到测试分类、各种测试技术、手动测试、测试自动化、测试报告、测试管理,以及怎样选择和使用自动化测试工具等,都进行了深入的探讨。本书的独到之处在于除介绍最新的软件测试科技外,还具有很高的实用价值。基于作者在国外从事一线软件测试的实践经验,书中介绍了这项工作所需的基础知识、测试架构的建立、测试操作的流程及测试实例。本书适合从事软件测试工作的专业人员和相关人员阅读与参考。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有,侵权必究。

图书在版编目(CIP)数据

实用软件测试:来自硅谷的技术、经验、心得和实例/李幸超著.

北京:电子工业出版社,2006.10

ISBN 7-121-03264-3

I.实... II.李... III.①软件-测试 IV.TP311.5

中国版本图书馆CIP数据核字(2006)第118351号

责任编辑:冯小贝

印 刷: 北京市牛山世兴印刷厂

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路173信箱 邮编:100036

开 本: 787×980 1/16 印张:13.5 字数:346千字

印 次:2006年10月第1次印刷

定 价:25.00元

凡所购买电子工业出版社的图书有缺损问题,请向购买书店调换;若书店售缺,请与本社发行部联系。联系电话:(010)68279077。邮购电话:(010)88254888。

质量投诉请发邮件至 zltts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

序

夜幕已临。“今天忘记了喝茶”，作者下班回家，进门便说。

这显然不是有益健康之道，但它反映出作者对所从事的软件测试工作的专注、认真以及背后所包含的热爱。此外，作者每天不停地学习和跟踪有关的最新科技知识和动向，而其所负责测试的软件已为许多政府部门、健康保险、银行等各种大机构所采用，而这些单位对软件质量的要求自然都是极高的。

这样一位多年来每天在美国硅谷软件公司从事实际软件测试工作的作者，以其经验、心得以及心血所写成的这本书，一定仍有疏漏，但也一定会对从事和有志于从事该行业的读者，特别是要了解相关技术的知识和现状并与国际接轨的读者有所帮助。

这应该就是作者以及作为第一读者的笔者最真挚的愿望。

旋永南^①

2006.8.30

^① 本序作者为 Flextronics (伟创力) 副总裁 / 高级 Fellow。

前 言

本书系统地介绍了有关软件测试的基本知识和全过程,从软件测试的规划和方案设计,到测试分类、各种测试技术、手动测试、测试自动化及测试报告、测试管理等,都进行了深入的探讨。书中还介绍了怎样选择和使用自动化测试工具及各种具体和详细的测试实例。此外,本书还特别给出了一些申请工作时可能遇到的面试题目,供有需要的读者参考。

第1章概述了软件测试,介绍了软件测试的基本概念、时机、对研发费用的影响、各阶段质量分析的要点等。本章还介绍了软件测试的分类、基本步骤,并探讨了美国硅谷地区软件公司的运作模式。

第2章介绍软件测试方案的制订与测试计划书,包括测试资料的收集与整理,方案的制订,计划书的编写及其要素和实例,测试记录,异常事件,以及缺陷报告和总报告,等等。此外,本章还讨论了测试文件的管理。

第3章简介了软件测试技术,内容包括白盒基本路径测试、渐进式测试及黑盒的功能测试和非功能测试。

第4章介绍了软件测试的进程。其中说明了软件开发与测试进度的关联(V模型)、单元测试、集成测试、系统测试、用户验收测试、回归测试的各种技术。

第5章介绍了各种软件测试的实例,包括用户界面与功能测试实例的编写、修改、增减和管理。

第6章说明了网络软件的测试,其中讨论了网络软件的组成部分及各种测试项目,包括用户界面、配置、兼容性、帮助系统、数据库、安全性、安装和卸装、软件性能等测试及强度测试。

第7章讲述了软件自动化测试的优缺点、测试工具(软件包)、步骤和程序的管理。

第8章给出了软件自动化测试的几个实例,涵盖了使用WinRunner、SilkTest、VisualTest等工具,以及在UNIX系统下的自动测试等各种情况。

附录A列出了一些软件测试工程师的面试题,它们大都来自美国硅谷软件公司实际使用的面试题目;附录B给出了实用测试计划书(样本)。最后,本书列出了一组参考文献,供有兴趣的读者进一步阅读。

本书作者长期在美国硅谷软件公司从事和负责软件测试工作,对本行业有着切身的体会,相信并希望本书对国内的同行能有所帮助。本书适合有一定计算机软件基础、希望学习最新软

件测试工程知识和技能的读者阅读,并可供已有工作经验的软件测试工程师参考。对于在外企任职或将在外企任职的软件测试工程师而言,本书可帮助他们实现知识更新并与国际接轨。此外,本书也适合作为软件测试岗位培训的教材。

旋采云小姐对本书的写作帮助良多,谨此致谢。

作者

2006年8月30日

于美国硅谷

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail : dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036

目 录

第 1 章 软件测试概述	1
1.1 为什么要对软件进行测试	1
1.2 何时应开始对软件进行测试	4
1.3 软件测试的分类	9
1.4 软件测试的基本步骤	19
1.5 美国硅谷地区软件公司的软件测试运作模式探讨	20
思考题	26
第 2 章 软件测试方案的制订与测试计划书的编写	27
2.1 测试资料的收集与整理	27
2.2 测试方案的制订	29
2.3 测试计划书的编写及其要素	30
2.4 实用测试计划书	41
思考题	42
第 3 章 软件测试技术简介	44
3.1 白盒测试与黑盒测试	44
3.2 白盒测试	44
3.3 黑盒测试	47
思考题	55
第 4 章 软件测试的阶段	56
4.1 V 模型——软件开发与测试进度的关联	56
4.2 单元测试	57
4.3 集成测试	60
4.4 系统测试	62
4.5 用户验收测试	63
4.6 回归测试	64
思考题	65

第 5 章 软件测试实例的编写	67
5.1 测试实例分类	67
5.2 用户界面测试实例的编写	68
5.3 功能测试实例的编写	72
5.4 测试实例的管理、改进与完善	99
思考题	104
第 6 章 网络软件测试	106
6.1 常用软件的结构模型	106
6.2 网络软件的组成部分	109
6.3 网络软件的测试项目	115
思考题	129
第 7 章 软件的自动化测试	131
7.1 自动化测试的优缺点	131
7.2 如何选择自动化测试软件 (工具)	133
7.3 测试自动化的步骤	137
7.4 自动测试程序的管理	139
第 8 章 软件自动化测试实例	140
8.1 动态网页的自动化测试 (使用 WinRunner)	140
8.2 网络软件的自动化测试 (使用 WinRunner 及 SilkTest)	144
8.3 窗口软件的自动化测试 (使用 WinRunner、SilkTest 及 VisualTest)	155
8.4 UNIX 系统下的自动测试	168
附录 A 软件测试工程师面试题目	186
附录 B 实用测试计划书 (样本)	195
参考文献	206

第1章 软件测试概述

1.1 为什么要对软件进行测试

在提倡科学管理、科学营商的今天,计算机软件设计也逐渐建立和形成了一套得到广泛认同的、科学的管理理论与方法,以达到提高产品质量、降低成本的目的。从事或准备进入这一行业的工程人员、管理人员,对软件测试有所认识并掌握相关技术是十分重要的。

质量是一切产品的生命线。与任何一种市场产品一样,软件业者当然希望自己的产品除了以功能方面的优势进军市场之外,还能以质量为立足市场之本。

作为软件的终端用户,哪怕价钱相对昂贵,人们也愿意购入已得到广泛应用、所占市场份额大、口碑好且出自财力雄厚的大公司的软件。这里面要考虑的因素很多,除了功能方面的要求之外,产品的质量是一个关键。即使一般软件都有售后服务,用户也必须慎重考虑花在获得售后服务上的时间、售后服务的质量及售后服务的保障是否长远。购买软件也是一种投资,软件是一件工具,谁也不希望买来的软件毛病多多;即使得到生产商的技术支持能解决问题,人们也不愿意把时间花在工具的修理上。这对购买、使用该软件的企业来讲是一种严重的伤害,除了资金付诸流水之外,时间上的浪费、人力资源的浪费和重新评估,购买新的软件所花费的时间、金钱,以及在使用问题软件的过程中业务上遭受的损失,都是难以挽回的。再者,软件是属于高科技领域的产品,常常影响重大,因此它的质量保证特别重要。此外,市场上的普通消费品如有一般的质量问题,消费者或许可以通过更换部件或找人维修得以继续使用;而软件一旦出现问题,只能通过推出该软件的软件公司予以修正,然后推出修正版,重新安装。这是因为终端用户获得的是可执行软件而不是源代码;而用户即使找到很有经验的编程专家,也无法自行纠错。

另一方面,从软件生产商的角度来看,软件的质量也是经营的命脉。从生产效率、售后服务到公司信誉,无一不与软件的质量紧密关联。其软件质量不高的生产商,在产品投放市场后,一定会被售后服务搞得焦头烂额,并造成经营成本飞涨;有的招架不住,只好撤下产品,不再提供售后服务;资本较弱的公司,干脆关闭了事。这对软件公司及用户来说,都是一种失败。因此,一个不争的事实是,在美国这样一个经济利益至上、软件工业十分发达且技术领先的营商环境里,软件测试在工业中占有举足轻重的位置。据笔者在美国硅谷地区工作的所见所闻,软件公司中聘用的软件程序员与测试员的比例从1:1~4:1不等,主要取决于产品消费群的大小、对象(像微软的Windows产品拥有大量的终端用户,其中包括非软件行业的用户,所以他们的测试人员比例很高),以及产品的复杂程度和产品所处的阶段(一般情况下,在新产

品将要推向市场前的一段时间会增加测试人手，以达到强化产品质量的目的)。随着产品不断地推出提升版，测试员对软件的了解加深，测试速度将不断提高。一旦测试实施自动化，对测试人员的需求就会下降。

软件的质量很重要，人们都同意这一点。但有人会问：程序员自己一边写程序一边测试不就行了吗，为什么要专门雇用测试人员呢？答案是：没错，如果程序员接受过测试的专门训练，那么他们也可以负责测试。不过，只要条件允许，设置专门的软件部门和机制是十分有益的。软件测试有自己的一套完整的、严格的理论与实践训练课程，测试人员工作的重点与程序员不一样，他们受到的训练当然也不一样。一个好的软件测试员必须具备的是：热爱自己的工作、对测试理论有深入的了解；掌握测试的各类技术，接受过专门的测试训练；对软件缺陷警觉性高；对缺陷可能出现的各种情况比较熟悉。与此同时，他们应掌握多种计算机语言，掌握一般软硬件知识、数据库知识及网络知识。更为重要的是，这些人员要对所要测试的软件的结构、功能及要求有深入的认知。

诚然，要在软件公司里专门设立测试部门，花费是不小的。除了财政方面的支出，还有时间上的付出。下面是设立测试部门时所需要解决的相关内容：

- 聘请经过严格训练的、有一定专业水平的、有经验的、热爱测试工作的测试工程师。
- 购置测试设备（包括计算机、测试软件等）。
- 测试人员培训（包括熟悉所要测试的软件的功能、操作；阅读该软件的相关文件，包括规格说明书、设计说明书等）。
- 测试系统的建立（包括测试数据的收集、测试计划的编写、测试实例的编写、测试自动化程序的编写等）。
- 测试具体操作。
- 报告缺陷。
- 回归测试。
- 不断修改完善测试实例及自动测试程序。

这一切看起来还真会让人犹豫，时间和金钱的付出是否值得？如果我们不去做，或者做得不够好，会有哪些可能出现的问题呢？首先，我们都会承认这样一个事实，那就是任何人编写的软件都会有缺陷。如果软件的缺陷能在内部测试时发现，那么缺陷修正的成本相对于从外部反馈回来的会低得多。其中的原因是，如果缺陷是在内部测试时发现的，编程部门接到报告后，可以安排相关程序员进行修正，并将修正后的新版本交回测试部门重新测试。相比之下，如果软件已经到了用户手中，终端用户在使用过程中发现缺陷并将之报告，那么除了修正及重测之外，软件公司还要通告所有现有用户重新下载修正版本或修补程序。粗看起来，这些处理好像都还是可以预计到的且成本有限；其实不然，如果软件的缺陷到了用户那里才被发现，那么软件公司也许会面临以下麻烦：

- 疲于应付各种预想不到的用户的埋怨。
- 公司形象受损，影响日后的销售。
- 如果缺陷造成用户方面的巨大经济损失，也许会吃上官司，赔偿数额不可预计。

这些情况，相信任何软件公司都不愿意见到。但是，尽管如此，并非所有软件公司都能意识到软件质量及建立独立测试部门的必要性；就算意识到了，也会抱着侥幸的心理，尝试着节省这笔开支。

上面列举的是软件公司在生产初期就应该意识到并应尽量避免其发生的不利局面。当然，这并不是说只要设立了测试部门、有一套比较完整的测试机制，就可以高枕无忧、麻烦不再来。不过可以肯定的是，这类麻烦出现的可能性会大大降低。

随着科学的进步，软件的应用扩展至商业、医疗、航空等领域。在银行服务软件中，一个软件缺陷有可能给银行或客户造成重大的损失；在机场，飞机调度软件缺陷会产生调度错误，将会造成机场的运作瘫痪……这些例子我们都听到过不少。在医疗系统、民航订票系统、证券交易系统、银行结算系统、军事防御系统及核电站安全控制系统等领域的软件应用，就更有必要把好质量关，因为这类应用关系重大，很可能造成灾难性的后果。2005年年初，eWeek就报道了这样一件事：在巴拿马国家癌症研究中心，自从2000年11月起，有27个癌症病人接受了超量伽马射线的照射。在之后的40个月里，有21个病人相继去世，而其中有5人的死因与接受了超量伽马射线的照射有直接的关系；另外15人因受到超量伽马射线的照射而引发了严重的并发症。这台放射仪器是由软件来控制的；经研究，认为这起医疗事故是由控制软件的缺陷及仪器操作技术员的失误所引起的。这个放射仪软件的出版商现正面临着来自本国及巴拿马的诉讼。其实早在1993年、1995年及1998年，美国国家食品及药品监督管理局（FDA）的监察员就已经注意到该公司没有很好的软件测试系统——没有足够的文件可以证明软件经过了严格的测试，放射量的计算有20%的误差，并且对客户的投诉没有认真处理，等等。而这些都说明了软件测试机制的重要性。以下所列是近年来发生的一些与软件缺陷有关的造成人命损失的故事：

- 2003年，软件问题造成美国东北部及加拿大停电，导致5000万人受影响，3人丧生。
- 2000年美国海军飞机坠落，导致4人丧生（控制软件问题）。
- 1997年韩国空难，导致225人丧生（雷达控制软件问题）。
- 1995年美国航空公司一架客机在哥伦比亚卡利机场降落时撞向一座山，调查认为是源于导航软件的问题。该事件导致159人丧生。

以上所列是与软件缺陷有关的、造成人命损失的故事。其他与软件缺陷有关的、造成经济损失的事件就更多了，以下是笔者搜集到的、近期的一些软件质量问题与经济损失的相关数字：

- 2004年，媒体报道过北美银行由于一个新安装的软件的缺陷，使得数以百万计的客户受到影响，该缺陷的修复花费了整整两个星期的时间，造成的损失以亿元计。

- 据报道, 2003 年 10 月, 美国的一个主要零售企业被政府重罚, 原因是其网站软件的缺陷使得客户相互间可以查看他人在该网站的订单。
- 2003 年 4 月, 美国一个专门为学生提供贷款的公司由于软件出错, 错误计算 80 万宗学生贷款利率, 导致了 800 万美元的利率损失。
- 前面提到的 2003 年由于软件问题造成美国东北部及加拿大历史上最大的停电事故, 各种损失估计约为 60 亿美元。

2002 年, 美国国家标准局发表的一份资料^①中说, 软件质量问题在美国每年造成 595 亿美元 (约占美国国民年生产总值的 0.6%) 的损失。其中软件公司方面的损失为 212 亿美元, 用户的损失为 383 亿美元。然而, 如果进一步提高软件的质量, 那么总的损失可以减少 222 亿美元, 其中软件公司可减少 106 亿美元的损失, 软件用户可减少损失 116 亿美元。

看到以上的数据, 确实很令人震撼。然而, 要保证软件的质量, 测试并非是唯一要考虑的重点, 同样重要的有

- 拥有一个素质高、懂行的高层领导小组。
- 拥有相关方面的专业人士 (建立软件的商业模型)、高水平的软件设计师 (定义软件系统、定义功能模块、定义各模块之间如何合作及各模块之间合作的接口)、高水平的软件编程工程师, 组成一支优秀的编程队伍。
- 有一个较好的软件系统设计。
- 公司高层充分重视软件质量控制, 预留足够的测试专用预算。
- 建立一套行之有效的科学管理运作系统, 明确各部门的分工与合作、明确对各职能的具体要求, 不断评估、不断改进及完善。

软件测试的目的是为了保证软件的最终质量。因此, 在软件开发的过程中, 测试部门必须不断对软件进行质量控制。但是, 我们也应该清醒地认识到, 测试只能证明软件程序有错, 而不能保证软件程序没有错。

1.2 何时应开始对软件进行测试

人们往往从一个小小的、现实生活中的需求开始, 酝酿出用软件的方法来协助满足该需求的想法。从一个想法到一个产品——一个拥有用户的、成熟的软件产品, 这个过程少说也要 2~3 年。即使在该软件推向市场后, 软件公司还是要对软件的各方面不断地进行改进, 包括功能、性能等, 也要不断推出新的版本。在这样一个漫长的过程中, 软件测试扮演着什么样的角色呢? 测试应该从哪里开始呢? 下面就让我们一起来探讨。

^① 以上数据的获取是基于对通信设备行业使用 CAD/CAE/PDM 软件和金融服务业软件的调查而得出的。

1.2.1 软件的生命周期

软件的生命周期（software life cycle）由以下几个主要过程组成（参见图 1.1）：

- 根据市场需求进行深入的调查研究，搜集相关资料。
- 对搜集到的资料进行分析，确定软件功能及性能要求。
- 软件整体设计。
- 软件编程设计。
- 软件编程及测试。
- 软件纠错 $\Leftrightarrow$ 测试。
- 软件推向市场。
- 不断修改提高……

软件生命周期示意图

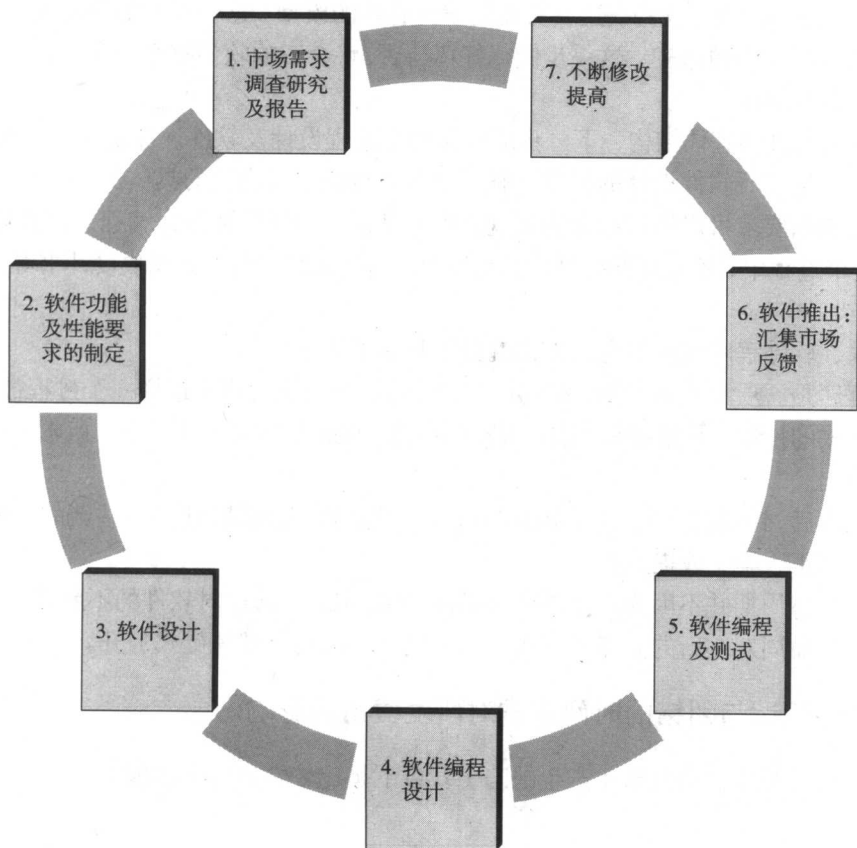


图 1.1 软件生命周期示意图

软件的生命周期始于市场的需求。软件商抓住市场的需求信息并迅速锁定客户群,进而了解企业或用户个人的特殊要求,最终确定软件的商业用途(如银行管理系统、图书管理系统、保密文件传输等)。

接下来的是软件具体功能的设定。这个过程要做的工作有不少是面对客户群的咨询,了解他们对软件的具体功能的需求。有些软件公司在这个过程中会做一个很简单的用户界面示范(往往用Visual Basic实现,因为它非常简单,并且随时可加入新的功能及改变原来的功能),让客户获得一个感性的认识。这个界面只是个示意的界面,不必有数据库或各种支持,不具备真正的功能,其目的只是为了收集客户对功能要求方面的意见。

在功能、性能要求确定以后,软件的规格说明书就应该定下来。接下来是软件的整体结构设计。这是软件生命周期中很关键的一环。这个阶段要做的工作包括分析、选定采用的语言、选定操作平台、选定数据库;软件如涉及网络通信,还要选择通信的协议与方式;等等。前面这些项都确定之后,还要为软件定义功能模块、各模块之间的联系、各模块之间如何合作,以及确定各模块之间的合作接口,这其实就是设计软件的模型。这一过程需要经过不断的评审,不断的改进,以及不断的完善。这一步做得好坏与否,将会影响软件整个生命周期中的每一环,因此是关键中的关键。

一旦确定软件的整体结构,下一步应该做的是编程设计及软件设计说明书的编写。

编程设计是按照软件设计的模型,制订一套可行的、最优化的具体流程,确保规格说明中的各项功能和要求可以通过编程去实现。也就是说,编程设计是为编程制订具体的路径及规范。在这个过程中,该软件的各种设计说明书(包括系统设计书、模块设计书等)也应同步完成。

接下来,软件程序编程的庞大工程就真正开始了。

编写程序是一项繁杂的工作。在此过程中,编程→测试→修改这样一个过程被不断重复,实现着螺旋式的推进。其结果是Alpha版本的推出、Beta版本的推出、销售版本(也称为黄金版本)的推出。

软件面市后,针对客户发现的问题或对软件功能的意见回馈(包括一些新的功能要求等),将不断推出修正版本(或提升版)。

软件的生命周期并不因为产品推向市场而终结;相反,通过对软件的不断修改、不断提高质量,这个生命周期在新的起点上再次开始、延续,从而呈现螺旋式上升。

1.2.2 软件测试与纠错的时间安排对研发费用的影响

首先,软件为什么会出错(产生缺陷)?以下是比较常见的一些因素:

- 软件模型拟定不正确（约 70% 的缺陷源于此，并且难以纠正）。
- 软件功能订立不明确或有错。
- 软件非常庞大，功能十分复杂。
- 编程过程出错（约 20% 缺陷源于此）。
- 个别功能要求改变而影响到其他部分。
- 第三方软件有缺陷。
- 人为因素（包括管理方法、时间太紧、人力资源不足、各部门之间或同事之间沟通不够等）。

在漫长的软件生命周期中，测试应从何时开始呢？不少学者做过许许多多的调查和研究，尽管各自获得的数据有异，而结论却是一致的：那就是越早进行测试，软件的总体研发费用反而越低。大致上说，一个初期就发现并修正的软件缺陷所需的费用，是一个已推向市场软件的缺陷修正费用的几十分之一。最新的数据显示（美国国家标准局 2002 年的调查结果）如图 1.2 所示^①。

软件所处阶段	功能需求搜集与分析 / 软件设计	编程 / 分块测试	整体 / 系统设计测试	早期用户反馈 / Beta 版	软件推出市场后
缺陷修正费用 / 耗时	1X	5X	10X	15X	30X

注：X 可视做单位耗时或单位费用。

图 1.2 软件修正费用

1.2.3 软件质量要点分析

软件测试和软件质量的概念是不可分离的。测试是保证质量的手段，质量是测试的目的。既然如此，我们对测试软件质量的评估有哪些要点呢？从哪些方面去审视测试中的软件呢？下面的几项即为考虑的重点：

- **完整性**（completeness）：软件是否具备软件规格说明书、设计说明书中所描述的各方面功能与性能。
- **正确性**（correctness）：软件的功能、性能是否达到设计规范中的设定。
- **可靠性**（reliability）：软件在正常使用的情况下，维持其功能及性能水准的程度。
- **兼容性**（compatibility）：软件在特定硬件、软件环境下正确运行及实现其功能的程度。
- **效率**（efficiency）：软件执行某项功能时所使用的计算机资源及时间的有效程度。

^① 笔者若干年前曾读过另一篇分析文章，结论数据更是达到 60X（上市后缺陷修正费用为早期修正费用的 60 倍），可惜当时没有记下文章的出处。图 1.2 是现在所能找到的最新调查结果，应该可以反映现实的情况。

- **可使用性** (usability): 用户学习、掌握该软件操作所要付出的时间及努力程度。
- **可移植性** (portability): 软件从一个系统或环境转移到另一系统或环境的难易程度。
- **可变比例性** (scalability): 软件在工作负荷持续增大的情况下其功能及性能指标保持的程度。
- **稳健性** (robustness): 软件在输入非正常数据或在单位时间内承受超强负荷的情况下其功能及性能指标保持的程度。
- **易测性** (testability): 软件在测试操作过程中所要付出的时间及努力程度。
- **可维护性** (maintainability): 当环境改变或软件发生错误时, 进行修正或恢复所做努力的程度。

最后的一项也是很重要的, 软件必须易于维护。用户在使用软件的过程中出现一般性的错误(如操作不当等引起的问题)时, 应能很快、很容易地得到解决, 而不需要求助于相关的专业人士。

1.2.4 软件成品阶段的质量要求分析

软件在其生命开始的那一刻起就在不断的检验、测试之中, 直到最后——也就是再没有人使用它的时候。

软件从成品的角度出发, 可分为三个阶段版本(参见图 1.3)。

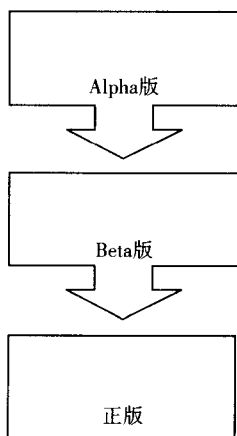


图 1.3 软件的三个阶段版本

1. Alpha 版

- 软件的所有功能已经基本实现, 可供内部使用。
- 所有的功能已通过测试, 一般情况下推向市场之前不再增减。