第 1 章　软件测试基础

本章目标

- 了解软件测试产生的背景。
- 掌握软件测试的定义。
- 熟悉软件测试的目的和原则。
- 了解软件测试的复杂性，并进行经济性分析。

软件测试是保证软件质量、提高软件可靠性的重要途径，软件测试的质量与测试人员的技能、经验以及对被测试软件的理解密切相关。随着时间的推移，软件测试的内涵在不断丰富，人们对软件测试的认识在不断深入。要完整理解软件测试，就要从不同角度去审视。

1.1　软件测试产生的背景

软件测试是伴随着软件的产生而产生的。在早期的软件开发过程中，软件规模较小，复杂程度低，软件开发的过程混乱无序且随意，此时软件测试的含义比较狭窄，开发人员将软件测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对软件测试的投入极少，软件测试的介入也较晚，常常是等到形成代码且产品已经基本完成时才进行测试。

直到 1957 年，软件测试才开始与调试区别开来，作为一种发现软件缺陷的活动。由于一直存在着“为了让我们看到产品在工作，就得将测试工作往后推一点”的思想，人们潜意识里对测试的目的就理解为“使自己确信产品能工作”。当时也缺乏有效的测试方法，主要依靠“错误推测”(Error Guessing)来寻找软件中的缺陷。测试活动始终晚于开发活动，软件测试通常被作为软件生命周期中最后一项活动而进行。因此，大量软件交付后，仍存在很多问题，软件产品的质量无法保证。

20 世纪 70 年代，这个阶段开发的软件仍然不复杂，但人们已开始思考软件开发流程的问题，尽管对软件测试的真正含义还缺乏共识，但这一词条已经频繁出现，一些软件测试的探索者们建议在软件生命周期的开始阶段就应该根据需求制订测试计划，这时也涌现出一批软件测试的宗师，Bill Hetzel 博士和 Glenford J. Myers 就是其中的领导者。

20 世纪 80 年代初期，软件和 IT 行业进入大发展时期，软件趋向大型化、高复杂度化，软件的质量越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，人们开

始为软件开发设计各种流程和管理方法,软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程,以结构化分析与设计、结构化评审、结构化程序设计以及结构化测试为特征。人们还将“质量”的概念融入其中,软件测试定义发生了改变,测试再不单纯是一个发现错误的过程,而是将测试作为软件质量保证(SQA)的主要职能,包含软件质量评价的内容。此时软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。因此,软件测试有了行业标准(IEEE/ANSI),软件测试成为一个专业,需要运用专门的方法和手段,需要专门的人才和专家来承担。

在竞争激烈的今天,无论是软件的开发商还是软件的使用者,都生存在竞争环境中。软件开发商为了占有市场,必须把产品质量作为企业的重要目标之一,以免在竞争中被淘汰出局。质量不佳的软件产品不仅会使开发上的维护费用和用户的使用成本大幅增加,还可能产生其他问题,造成公司信誉下降。一些关键的应用领域如果质量有问题,还可能造成灾难性的后果。现在人们已经逐步认识到软件中存在的错误会导致软件开发在成本、进度和质量上的失控。由于软件是由人设计完成的,所以它不可能十全十美,虽然不可能完全杜绝软件中的错误,但是可以用软件测试等手段使程序中的错误数量尽可能少,密度尽可能小。

国际上,软件测试(软件质量控制)是一件非常重要的工程工作,测试也作为一个非常独立的职业而存在。在IBM、Microsoft等开发大型系统的软件公司中,很多重要项目的开发中的测试人员与软件开发人员的比例能够达到1∶2或1∶4。在软件测试技术方面,自动化测试系统(ATS)正朝着通用化、标准化、网络化和智能化的方向迈进。20世纪90年代中期以来,自动测试系统开发研制的指导思想发生了重大变化,以综合通用的ATS代替某一系列,采用共同的硬件及软件平台实现资源共享的思想受到高度重视。其主要思路是:采用共同的测试策略,从设计过程开始,通过“增值开发”的方式使后一阶段测试设备的研制能利用前一阶段的开发成果;TPS要能够移植,软件模块可以重用;使用商业通用标准、成熟的仪器设备,缩短研发时间,降低开发成本并且易于升级和扩展。

国内软件测试的现状主要表现在如下几点。

(1) 软件测试的地位不高,在很多公司还是一种可有可无的技术,大多只停留在软件单元测试、集成测试和功能测试上。

(2) 软件测试标准化和规范化不够。

(3) 软件测试从业人员的数量同实际需求有不小的差距,国内软件企业中开发人员与测试人员数量一般为5∶1,国外一般为2∶1或1∶1,而最近有资料显示Microsoft已把此比例调整为1∶2。

(4) 国内缺乏完全商业化的操作机构,一般只是政府部门的下属机构在做一些产品的验收测试工作,实质意义不大,软件测试产业化还有待开发和深掘。

因此,我国的软件测试行业较欧美国家的差距还比较大。通过研究发现,造成这种情况的原因主要有如下几点。

(1) 国内软件产业本身不强大,软件质量较低。

(2) 软件管理者与用户对软件质量的认识有待加强。

(3) 软件管理者对软件测试的认识和重视程度不够。

(4) 软件行业质量监督体系不够好。

(5) 软件从业人员的素质不够高。

(6) 软件测试行业处于起步阶段,经济效益短期内不明显。

1.2　软件测试的定义

1983 年 IEEE 提出的软件工程术语中给软件测试下的定义是:“软件测试是使用人工或自动的手段来运行或测定某个软件系统的过程,其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。”这个定义明确指出:软件测试的目的是为了检验软件系统是否满足需求;软件测试是贯穿于整个开发流程的。

其扩展定义:软件测试就是在软件投入运行前,对软件需求分析、设计规格说明和编码的最终复审,是软件质量保证的关键步骤。

软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例(包括输入数据与预期输出结果),并利用这些测试用例运行软件,以发现软件错误的过程。广义的软件测试是由确认、验证、测试 3 个方面组成。

(1) 确认是指评估将要开发的软件产品是否准确无误,是否可行和有价值。确认意味着确保一个待开发软件是准确无误的,是对软件开发构想的检测;主要体现在计划阶段、需求分析阶段,也会出现在测试阶段。

(2) 验证是指检测软件开发的每个阶段、每个步骤的结果是否准确无误,是否与软件开发各阶段的要求或期望的结果相一致。验证意味着确保软件会准确无误地实现软件的需求,开发过程是沿着正确的方向进行的,主要体现在设计阶段、编码阶段。

(3) 测试是指它与狭隘的测试概念统一,主要体现在编码阶段和测试阶段。

确认、验证与测试是相辅相成的。确认产生验证和测试的标准,验证和测试帮助完成确认。

1.3　软件测试的目的

软件测试的目的是利用有限的资源找出对用户影响最深的缺陷(Bug)。不同的机构会有不同的测试目的;相同的机构也可能有不同的测试目的,可能是测试不同区域或是对同一区域的不同层次的测试。测试目的决定了如何去组织测试。如果测试的目的是为了尽可能多地找出错误,那么测试就应该直接针对软件比较复杂的部分或是以前出错比较多的位置。如果测试目的是为了给最终用户提供具有一定可信度的质量评价,那么测试就应该直接针对在实际应用中会经常用到的商业假设。

在谈到软件测试时,许多人都引用 Grenford J. Myers 在 *The Art of Software Testing* 一书中的观点。

(1) 软件测试是为了发现错误而执行程序的过程。

(2) 测试是为了证明程序有错,而不是证明程序无错误。

(3) 一个好的测试用例在于它能发现至今未发现的错误。

(4) 一个成功的测试是发现了至今未发现的错误的测试。

这种观点可以提醒人们测试要以查找错误为中心,而不是为了演示软件的正确功能。但是仅凭字面意思理解这一观点可能会产生误导,认为发现错误是软件测试的唯一目的,查找不出错误的测试就是没有价值的,事实并非如此。

首先,测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特征,可以帮助项目管理者发现当前所采用的软件过程的缺陷,以便改进。同时,这种分析也能帮助我们设计出有针对性的检测方法,从而改善测试的有效性。

其次,没有发现错误的测试也是有价值的,完整的测试是评定测试质量的一种方法。详细而严谨的可靠性增长模型可以证明这一点。例如,Bev Littlewood 发现一个经过测试而正常运行了若干小时的系统,有可能继续正常运行很多小时。

1.4 软件测试的原则

基于软件测试是为了寻找软件的错误与缺陷,评估与提高软件质量,我们提出如下测试原则。

1. 所有的软件测试都应追溯到用户需求

这是因为软件的目的是使用户完成预定的任务,并满足用户的需求,而软件测试所揭示的缺陷和错误使软件达不到用户的目标,满足不了用户需求。

2. 应当把"尽早地和不断地进行软件测试"作为软件测试者的座右铭

由于软件的复杂性和抽象性,在软件生命周期各个阶段都可能产生错误,所以不应把软件测试仅仅看作是软件开发的一个独立阶段的工作,而应当把它贯穿到软件开发的各个阶段,并且在软件开发的需求分析和设计阶段就进行测试工作,编写测试文档,这样才能在开发过程中尽早发现和预防错误,杜绝某些缺陷和隐患,从而提高软件质量。问题发现得越早,解决问题的代价就越小,这算是一条真理。发现软件错误的时间在整个软件过程阶段越靠后,修复它所消耗的资源就越大,如图 1-1 所示。

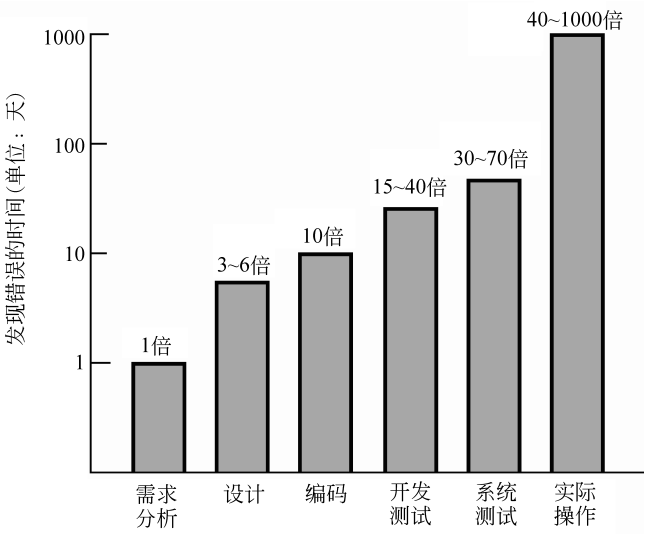

图 1-1 修改一个错误的相对成本

3. 完全测试是不可能的，测试需要终止

在测试中，由于输入量太大、输出结果太多，以及路径组合太多，想要进行完全的测试，在有限的时间和资源条件下几乎是不可能的。下面我们以大家所熟悉的计算器为例来说明，如图 1-2 所示。

图 1-2　计算器

输入：1＋0、1＋1、…、1＋9…9，全部完成后继续操作 2＋1、2＋2、…、2＋9，全部整数完成，则开始测试小数 1.0＋1.1、1.0＋1.2…，一直持续下去。

在验证完整数相加、小数相加后，继续进行后面的减、乘、除运算。我们还需要测试一下可能的错误输入，比如 1＋"、2＋!、3＋@、＃＋＄……这些组合有千千万万种。

4. 测试无法显示软件潜在的缺陷

进行测试可以查找并报告所发现的软件缺陷和错误，但不能保证软件的缺陷和错误全部找到，继续进一步测试可能还会找到一些。也就是说，测试只能证明软件存在错误而不能证明软件没有错误。换句话说，彻底的测试是不可能的。

5. 充分注意测试汇总的群集现象

经验表明，测试后的程序中残存的错误数目与该程序中已发现的错误数目或检错率成正比。根据这个规律，我们要对错误群集的程序段进行重点测试，以提高测试投资的有效率。例如，在美国 IBM 公司的 OS/370 操作系统中，47％的错误仅与该系统的 4％的程序模块有关。

6. 程序员应避免检查自己的程序

从心理上来说，人们总不愿承认自己有错，而让程序员来揭示自己的错误也比较难，因此，为了达到测试目的，我们尽量让单独的测试部门来做。

7. 尽量避免测试的随意性

测试是一种有组织、有计划、有步骤的活动，不是随意的工作。

1.5　软件测试的复杂性与经济性分析

人们在对软件工程开发的常规认识中，认为开发程序是一个复杂而困难的过程，需要花费大量的人力、物力和时间，而测试一个程序则比较容易，不需要花费太多的精力。这其实是人们对软件工程开发过程理解上的一个误区。在实际的软件开发过程中，作为现代软件开发工业一个非常重要的组成部分，软件测试正扮演着越来越重要的角色。随着软件规模的不断扩大，如何在有限的条件下对被开发软件进行有效的测试正成为软件工程中一个非常关键的课题。

设计测试用例是一项细致并且需要具备高度技巧的工作，稍有不慎就会顾此失彼，发生不应有的疏漏。下面分析一下容易出现问题的根源。

1. 完全测试是不现实的

在实际的软件测试工作中，不论采用什么方法，由于软件测试工作量非常大，不可能进行完全彻底的测试。所谓彻底测试，就是让被测程序在一切可能的输入情况下全部执行一

遍。通常也称这种测试为穷举测试。

穷举测试会出现如下几个问题。

(1) 输入量太大。

(2) 输出结果太多。

(3) 软件执行的路径太多。

(4) 说明书存在主观性。

E. W. Dijkstra 的一句名言对测试的不彻底性做了很好的注解:“程序测试只能证明错误的存在,但不能证明错误的不存在。”由于穷举测试工作量太大,实际操作时行不通,这就注定了一切实际测试都是不彻底的,也就不能够保证被测试程序在理论上不存在遗留的错误。

2. 软件测试是有风险的

穷举测试的不可行性使得大多数软件在进行测试的时候只能采取非穷举测试,这又意味着一种冒险。比如,在使用 Microsoft Office 工具中的 Word 时,可以做这样的一个测试:①新建一个 Word 文档;②在文档中输入汉字“胡”;③设置其字体为“隶书”,字号为“初号”,效果为“空心”;④将页面的显示比例设为 500%。这时在“胡”字的内部会出现“胡万进印”4 个字。类似问题在实际测试中如果不使用穷举测试是很难发现的,而如果在软件投入市场后才发现,则修复代价会非常高。这就会产生一个矛盾:软件测试员不能做到完全测试,不完全测试又不能证明软件百分之百可靠,那么如何在这两者的矛盾中找到一个相对的平衡点呢?

如图 1-3 所示,当软件缺陷降低到某一数值后,随着测试工作量的不断上升,软件缺陷并没有明显地下降。这是软件测试工作中需要注意的重要问题。如何把测试数据量巨大的软件测试减少到可以控制的范围,如何针对风险做出最明智的选择,是软件测试人员必须要把握的关键问题。

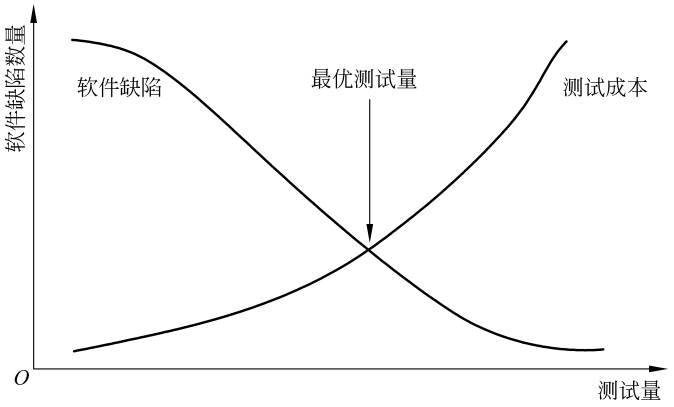

图 1-3　最优测试量示意图

最优测试量示意图说明了发现软件缺陷数量和测试量之间的关系,随着测试量的增加,测试成本将呈几何数级上升,而软件缺陷降低到某一数值之后将没有明显的变化,最优测量值就是这两条曲线的交点。

3. 杀虫剂现象

1990 年，Boris Beizer 在其编著的 *Software Testing Techniques*（*2nd Edition*）中提到了“杀虫剂怪事”一词，其代表的意思是同一种测试工具或方法用于测试同一类软件越多，则被测试软件对测试的免疫力就越强。这与农药杀虫是一样的道理，因为总是用一种农药，所以害虫就有了免疫力，农药就失去了作用。

由于软件开发人员在开发过程中可能会遇见各种各样的问题，再加上不可预见的突发性问题，所以即使优秀的软件测试员也不可能检测出软件中所有的缺陷。为了克服被测试软件的免疫力，软件测试员必须不断地编写新的测试程序，且对程序的各个部分进行不断地测试，以避免被测试软件对单一的测试程序具有免疫力而使软件缺陷不被发现。这就对软件测试人员的素质提出了很高的要求。

4. 缺陷的不确定性

在软件测试中还有一个让人不容易判断的现象是缺陷的不确定性，即并不是所有的软件缺陷都需要被修复。那么究竟什么才算是软件缺陷？这是一个很难把握的标准，在任何一本软件测试的书中都只能给出一个笼统的定义。在实际测试中，需要把这一定义根据具体的被测对象明确化。即使这样，具体的测试人员对软件系统的理解不同，还是会出现不同的标准。

软件测试的经济性具有如下两方面。

（1）体现在测试工作在整个项目开发过程中的重要地位。

（2）体现在应该按照什么样的原则进行测试，以实现测试成本与测试效果的统一。

1.6　本章小结

本章主要介绍了软件测试产生的背景，软件测试的定义、目的、原则，以及软件测试的复杂性与经济性分析。软件测试是伴随着软件的产生而产生的。早期的软件开发过程中，软件规模都很小，复杂程度低，软件开发的过程混乱无序且随意，测试的含义比较狭窄，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的错误。软件测试就是为发现缺陷而运行程序的过程。广义的软件测试是由确认、验证、测试 3 个方面组成。软件测试的目的是为了尽早发现软件系统中的缺陷，对缺陷进行跟踪管理，确保每个被发现的缺陷都能够及时得到处理。

1.7　练习题

1. 判断题

（1）验证意味着确保软件准确无误地实现软件的需求，开发过程是沿着正确的方向进行。（　　）

（2）调试的目的是发现缺陷。（　　）

（3）软件缺陷主要来自产品说明书的编写和产品方案设计。（　　）

(4) 在实际的软件测试工作中,不论采用什么方法,由于软件测试情况数量极其巨大,都不可能进行完全彻底的测试。 (　　)

(5) 测试人员可以不懂编程。 (　　)

2. 选择题

(1) 软件是程序和(　　)的集合。

A. 代码　　B. 文档　　C. 测试用例　　D. 测试

(2) 严重的软件缺陷的产生主要源自(　　)。

A. 需求　　B. 设计　　C. 编码　　D. 测试

(3) Fixed 的意思是指(　　)。

A. 该 bug 没有被修复,并且得到了测试人员的确认

B. 该 bug 被拒绝了,并且得到了测试人员的确认

C. 该 bug 被修复了,并且得到了测试人员的确认

D. 该 bug 被关闭了,并且得到了测试人员的确认

(4) 降低缺陷费用最有效的方法是(　　)。

A. 测试尽可能全面　　B. 尽可能早地开始测试

C. 测试尽可能深入　　D. 让用户进行测试

(5) 以下不属于应用系统中的缺陷类型的是(　　)。

A. 不恰当的需求解释　　B. 用户指定的错误需求

C. 设计人员的习惯不好　　D. 不正确的程序规格说明

3. 简答题

(1) 请简述软件缺陷包含了哪些内容。

(2) 请简述软件测试的定义。