第 1 章

软件测试概述

本章概述

本章介绍了软件测试的发展历史，软件测试技术的分类方法、测试标准、测试原则，阐述了软件测试与软件开发的关系。

1.1 软件测试的背景

软件的质量就是软件的生命，为了保证软件的质量，人们在长期的开发过程中积累了许多经验并形成了许多行之有效的方法。但是借助这些方法，我们只能尽量减少软件中的错误和不足，却不能完全避免所有的错误。

由于软件是人脑的高度智力化的体现和产品这一特殊性，不同于其他科技和生产领域，因此软件与生俱来就有可能存在着缺陷。

在开发大型软件系统的漫长过程中，面对纷繁复杂的各种现实情况，人的主观认识和客观现实之间往往存在着差距，开发过程中的各类人员之间的交流和配合也往往并不是尽善尽美的。

如果我们不能在软件正式投入运行之前发现并纠正这些错误，那么这些错误最终必然会在软件的实际运行过程中暴露出来。到那时，不仅改正这些错误要付出很大的代价，而且往往会造成无法弥补的损失。

如何防止和减少这些可能存在的问题呢？回答是进行软件测试。测试是最有效的排除和防止软件缺陷与故障的手段，并由此促进了软件测试理论与技术实践的快速发展。新的测试理论、测试方法、测试技术手段在不断涌出，软件测试机构和组织也在迅速产生和发展，由此软件测试技术职业也同步完善和健全起来。

1.1.1 软件缺陷

1. 软件错误案例研究

人们常常不把软件当回事，没有真正意识到它已经深入渗透到我们的日常生活中，软件在电子信息领域里无处不在。现在有许多人如果一天不上网查看电子邮件，简直就没法过下去。我们已经离不开 24 小时包裹投递服务、长途电话服务和最先进的医疗服务了。

然而软件是由人编写开发的，是一种逻辑思维的产品，尽管现在软件开发者采取了一系列有效措施，不断地提高软件开发质量，但仍然无法完全避免软件(产品)会存在各种各样的缺陷。

下面以实例来说明。

(1) 迪斯尼的狮子王游戏软件缺陷

1994 年秋天，迪斯尼公司发布了第一个面向儿童的多媒体光盘游戏——狮子王动画故事书(The Lion King Animated Storybook)。尽管当时已经有许多其他公司在儿童游戏市场上运作多年，但是这次是迪斯尼公司首次进军儿童游戏市场，所以进行了大量促销宣传。结果，销售额非常可观，该游戏成为孩子们在当年节假日的“必买游戏”。然而后来却飞来横祸。1994 年 12 月 26 日，圣诞节过后的第一天，迪斯尼公司的客户支持电话开始响个不停。很快，电话支持技术员们就淹没在来自于愤怒的家长并伴随着玩不成游戏的孩子们哭叫的电话之中。报纸和电视新闻进行了大量的报道。

后来证实，迪斯尼公司未能对市面上投入使用的许多不同类型的 PC 机型进行广泛的测试。软件只能在极少数系统中工作正常——例如在迪斯尼程序员用来开发游戏的系统中——但在大多数公众使用的系统中却不能运行。

(2) 爱国者导弹防御系统缺陷

爱国者导弹防御系统是里根总统提出的战略防御计划(即星球大战计划)的缩略版本，它首次应用在海湾战争中对抗伊拉克飞毛腿导弹的防御战中。尽管对系统赞誉的报道不绝于耳，但是它确实在对抗几枚导弹中失利，包括一次在沙特阿拉伯的多哈击毙了 28 名美国士兵。分析发现症结在于一个软件缺陷，系统时钟的一个很小的计时错误积累起来到 14 小时后，跟踪系统不再准确。在多哈的这次袭击中，系统已经运行了 100 多个小时。

(3) 千年虫问题

20 世纪 70 年代早期的某个时间，某位程序员正在为本公司设计开发工资系统。他使用的计算机存储空间很小，迫使他尽量节省每一个字节。他将自己的程序压缩得比其他任何人都紧凑。使用的其中一个方法是把 4 位数年份，例如 1973 年，缩减为 2 位数，73。因为工资系统相当依赖于日期的处理，所以需要节省大量的存储空间。他简单地认

为只有在到达 2000 年，那时他的程序开始计算 00 或 01 这样的年份时问题才会产生。虽然他知道会出这样的问题，但是他认定在 25 年之内程序肯定会升级或替换，而且眼前的任务比现在计划遥不可及的未来更加重要。然而这一天毕竟到来了。1995 年他的程序仍然在使用，而他退休了，谁也不会想到如何深入到程序中检查 2000 年兼容问题，更不用说去修改了。

估计全球各地更换或升级类似的前者程序以解决潜在的 2000 年问题的费用已经达数千亿美元。

(4) 美国航天局火星登陆探测器缺陷

1999 年 12 月 3 日，美国航天局的火星极地登陆者号探测器试图在火星表面着陆时失踪。一个故障评估委员会调查了故障，认定出现故障的原因极可能是一个数据位被意外置位。最令人震惊的问题是为什么没有在内部测试时发现呢。

从理论上看，着陆的计划是这样的：当探测器向火星表面降落时，它将打开降落伞减缓探测器的下降速度。降落伞打开几秒钟后，探测器的三条腿将迅速撑开，并锁定位置，准备着陆。当探测器离地面 1800 米时，它将丢弃降落伞，点燃着陆推进器，缓缓地降落到地面。

美国航天局为了省钱，简化了确定何时关闭着陆推进器的装置。为了替代其他太空船上使用的贵重雷达，他们在探测器的脚部装了一个廉价的触点开关，在计算机中设置一个数据位来控制触点开关关闭燃料。很简单，探测器的发动机需要一直点火工作，直到脚“着地”为止。

遗憾的是，故障评估委员会在测试中发现，许多情况下，当探测器的脚迅速撑开准备着陆时，机械震动也会触发着陆触点开关，设置致命的错误数据位。设想探测器开始着陆时，计算机极有可能关闭着陆推进器，这样火星极地登陆者号探测器飞船下坠 1800 米之后冲向地面，撞成碎片。

结果是灾难性的，但背后的原因却很简单。登陆探测器经过了多个小组测试。其中一个小组测试飞船的脚折叠过程，另一个小组测试此后的着陆过程。前一个小组不去注意着陆数据是否置位——这不是他们负责的范围；后一个小组总是在开始复位之前复位计算机，清除数据位。双方独立工作都做得很好，但合在一起就不是这样了。

(5) 金山词霸缺陷

在国内，“金山词霸”是一个很著名的词典软件，应用范围极大，对使用中文操作的用户帮助很大，但它也存在不少缺陷。例如输入“cube”，词霸会在示例中显示 $3^3=9$ 的错误；又如，如果用鼠标取词“dynamically”(力学，动力学)，词霸会出现其他不同的单词“dynamite *n.* 炸药”的显示错误。

(6) 英特尔奔腾浮点除法缺陷

在计算机的“计算器”程序中输入以下算式：

(4195835/3145727) * 3145727－4195835

如果答案是0,就说明计算机没问题。如果得出别的结果,就表示计算机使用的是带有浮点除法软件缺陷的老式英特尔奔腾处理器——这个软件缺陷被烧录在一个计算机芯片中,并在制作过程中反复生产。

1994年10月30日,弗吉尼亚州Lynchburg学院的Thomas R. Nicely博士在他的一个实验中,用奔腾PC机解决一个除法问题时,记录了一个想不到的结果,得出了错误的结论。他把发现的问题放到因特网上,随后引发了一场风暴,成千上万的人发现了同样的问题,并且发现在另外一些情形下也会得出错误的结果。万幸的是,这种情况很少见,仅仅在进行精度要求很高的数学、科学和工程计算中才会导致错误。大多数用来进行税务处理和商务应用的用户根本不会遇到此类问题。

这件事情引人关注的并不是这个软件缺陷,而是英特尔公司解决问题的方式:

- 他们的软件测试工程师在芯片发布之前进行内部测试时已经发现了这个问题。英特尔的管理层认为这没有严重到要保证修正,甚至公开的程度。
- 当软件缺陷被发现时,英特尔通过新闻发布和公开声明试图弱化这个问题的已知严重性。
- 受到压力时,英特尔承诺更换有问题的芯片,但要求用户必须证明自己受到缺陷的影响。

舆论哗然。互联网新闻组里充斥着愤怒的客户要求英特尔解决问题的呼声。新闻报道把英特尔公司描绘成不关心客户和缺乏诚信者。最后,英特尔为自己处理软件缺陷的行为道歉并拿出4亿多美元来支付更换问题芯片的费用。现在英特尔在Web站点上报告已发现的问题,并认真查看客户在互联网新闻组里所留下的反馈意见。

2. 软件缺陷的定义

从上述的案例中可以看到软件发生错误时将造成灾难性危害或对用户产生各种影响。

在这些事件中,显然软件未按预期目标运行。作为软件测试员,可能会发现大多数缺陷不如上面所列举的实例那么明显,而对于一些简单而细微的错误,很难做到真正区分哪些是真正的错误,哪些不是。对于软件存在的各种问题我们都称为软件缺陷或软件故障。在英文中人们喜欢用一个不贴切但已经专用的词“bug”表示。

软件缺陷即计算机系统或者程序中存在的任何一种破坏正常运行能力的问题、错误,或者隐藏的功能缺陷、瑕疵。缺陷会导致软件产品在某种程度上不能满足用户的需要。对于软件缺陷的准确定义,通常有以下5条描述:

① 软件未实现产品说明书要求的功能。

② 软件出现了产品说明书指明不会出现的错误。

③ 软件实现了产品说明书未提到的功能。

④ 软件实现了产品说明书虽未明确指出但应该实现的目标。

⑤ 软件难以理解，不易使用，运行缓慢或者终端用户认为不好。

为了更好地理解每一条规则，我们以计算器为例进行说明。

计算器的产品说明书声称它能够准确无误地进行加、减、乘、除运算。当你拿到计算器后，按下(＋)键，结果什么反应也没有，根据第①条规则，这是一个缺陷。假如得到错误答案，根据第①条规则，这同样是一个缺陷。

若产品说明书声称计算器永远不会崩溃、锁死或者停止反应。当你任意敲键盘，计算器停止接受输入，根据第②条规则，这是一个缺陷。

若用计算器进行测试，发现除了加、减、乘、除之外它还可以求平方根，说明书中从没提到这一功能，根据第③条规则，这是软件缺陷。软件实现了产品说明书未提到的功能。

若在测试计算器时，发现电池没电会导致计算不正确，但产品说明书未指出这个问题。根据第④条规则，这是个缺陷。

第⑤条规则是全面的。如果软件测试员发现某些地方不对劲，无论什么原因，都要认定为缺陷。如“＝”键布置的位置极其不好按；或在明亮光下显示屏难以看清。根据第⑤条规则，这些都是缺陷。

美国商务部国家标准和技术研究所(NIST)进行的一项研究表明，软件中的 bug 每年给美国经济造成的损失高达 595 亿美元。说明软件中存在的缺陷所造成的损失是巨大的，从反面又一次证明软件测试的重要性。如何尽早彻底地发现软件中存在的缺陷是一项非常复杂，需要创造性和高度智慧的工作。同时，软件的缺陷是软件开发过程中的重要属性，反映软件开发过程中需求分析、功能设计、用户界面设计、编程等环节所隐含的问题，也为项目管理、过程改造提供了许多信息。

3. 软件缺陷的原因

软件缺陷的产生，首先是不可避免的。其次，我们可以从软件本身，团队工作和技术问题等多个方面分析，将比较容易确定造成软件缺陷的原因归纳如下。

(1) 技术问题

- 算法错误。
- 语法错误。
- 计算和精度问题。
- 系统结构不合理，造成系统性能问题。
- 接口参数不匹配出现问题。

(2) 团队工作

- 系统分析时对客户的需求不是十分清楚，或者和用户的沟通存在一些困难。
- 不同阶段的开发人员相互理解不一致，软件设计对需求分析结果的理解偏差，编

程人员对系统设计规格说明书中某些内容重视不够，或存在着误解。

- 设计或编程上的一些假定或依赖性，没有得到充分地沟通。

(3) 软件本身

- 文档错误、内容不正确或拼写错误。
- 数据考虑不周全引起强度或负载问题。
- 对边界考虑不够周全，漏掉某几个边界条件造成的错误。
- 对一些实时应用系统，保证精确的时间同步，否则容易引起时间上不协调、不一致性带来的问题。
- 没有考虑系统崩溃后在系统安全性、可靠性的隐患。
- 硬件或系统软件上存在的错误。
- 软件开发标准或过程上的错误。

4. 软件缺陷的组成

我们知道软件缺陷是由很多原因造成的，如果把它们按需求分析结果——规格说明书、系统设计结果、编程的代码等归类起来，比较后发现，规格说明书是软件缺陷出现最多的地方，如图 1-1 所示。

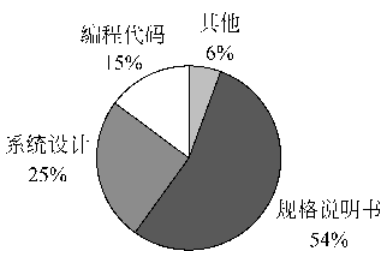

图 1-1 软件缺陷构成示意图

软件产品规格说明书为什么是软件缺陷存在最多的地方，主要原因有以下几种。

(1) 用户一般是非计算机专业人员，软件开发人员和用户的沟通存在较大困难，对要开发的产品功能理解不一致。

(2) 由于软件产品还没有设计、开发，完全靠想象去描述系统的实现结果，所以有些特性还不够清晰。

(3) 需求变化的不一致性。用户的需求总是在不断变化的，这些变化如果没有在产品规格说明书中得到正确的描述，容易引起前后文，上下文的矛盾。

(4) 对规格说明书不够重视，在规格说明书的设计和写作上投入的人力、时间不足。

(5) 没有在整个开发队伍中进行充分沟通，有时只有设计师或项目经理得到比较多的信息。

5. 软件缺陷的修复费用

软件不仅仅是表面上的那些东西——通常要靠有计划、有条理的开发过程来实现。从开始到计划、编程、测试，到公开使用的过程中，都有可能发现软件缺陷。

软件缺陷造成的修复费用呈指数级地增长——也就是说，随着时间的推移，由于软件缺陷造成的修复费用呈十倍地增长。当早期编写产品说明书时发现并修复缺陷，费用只

要 1 美元甚至更少。同样的缺陷如果直到软件编写完成开始测试时才发现，费用可能要 10～100 美元。如果是客户发现的，费用可能达到数千甚至数百万美元。

举一个例子来说明，比如前面的迪斯尼狮子王实例。问题的根本原因是软件无法在流行的 PC 平台上运行。假如早在编写产品说明书时，有人已经研究过什么 PC 机流行，并且明确指出软件需要在该种配置上设计和测试，付出的代价小得几乎可以忽略不计。如果没有这样做，还有一个补救措施是，软件测试员去搜集流行 PC 样机并在其上验证。他们可能会发现软件缺陷，但是修复费用要高得多，因为软件必须调试、修改、再测试。开发小组还应当把软件的初期版本分发给一小部分客户进行试用 这叫 Beta 测试。那些被挑选出来代表庞大市场的客户可能会发现问题。然而实际的情况是，缺陷被完全忽视，直到成千上万的光盘被压制和销售出去。而迪斯尼公司最终支付了客户投诉电话费、产品召回、更换光盘，以及又一轮调试、修改和测试的费用。如果严重的软件缺陷到了客户那里，就足以耗尽整个产品的利润。

1.1.2　软件测试技术的发展历史和现状

1. 软件测试技术的发展历史

随着计算机的诞生——在软件行业发展初期就已经开始实施软件测试，但这一阶段还没有系统意义上的软件测试，更多的是一种类似调试的测试。测试是没有计划和方法的，测试用例的设计和选取也都是根据测试人员的经验随机进行的，大多数测试的目的是为了证明系统可以正常运行。

20 世纪 50 年代后期到 20 世纪 60 年代，随着计算机软件的发展，用于计算机编程的各种高级语言也相继诞生，测试的重点也逐步转入到使用高级语言编写的软件系统中来，但程序的复杂性远远超过了以前。尽管如此，由于受到硬件的制约，在计算机系统中，软件仍然处于次要位置。软件正确性的把握仍然主要依赖于编程人员的技术水平。因此，这一时期软件测试的理论和方法发展比较缓慢。

20 世纪 70 年代以后，随着计算机处理速度的提高，存储器容量的快速增加，软件在整个计算机系统中的地位变得越来越重要。随着软件开发技术的成熟和完善，软件的规模也越来越大，复杂度也大大增加。因此，软件的可靠性面临着前所未有的危机，给软件测试工作带来了更大的挑战，很多测试理论和测试方法应运而生，逐渐形成了一套完整的体系，培养和造就了一批批出色的软件测试人才。

如今在软件产业化发展的大趋势下，人们对软件质量、成本和进度的要求也越来越高，软件质量的控制已经不仅仅是传统意义上的软件测试。传统软件的测试大多是基于代码运行的，并且常常是软件开发的后期才开始进行，但大量研究表明，设计活动引入的错误占软件开发过程中出现的所有错误数量的 50％～65％。因此，越来越多的声音呼吁，要求有一个规范的软件开发过程。而在整个软件开发过程中，测试已经不再只是基于

程序代码进行的活动，而是一个基于整个软件生命周期的质量控制活动，贯穿于软件开发的各个阶段。

2. 软件测试的现状

在我国，软件测试可能算不上一个真正的产业，软件开发企业对软件测试认识淡薄，软件测试人员与软件开发人员往往比例失调，而在发达国家和地区软件测试已经成为了一个产业，微软的开发工程师与测试工程师的比例是 1∶2，国内一般公司是 6∶1。很多人认为导致这种现状产生的原因是与我们接受的传统教育和开发习惯有相当大的关系。软件行业相对于其他一些行业来说是相当年轻的，开发过程包含了需求分析、设计、编程、测试和维护等工作，由于软件业的历史年轻，而且一般人认为，开发周期前面的工作没有完善之前，比较难于考虑到后面的工作。因此，我们可以看到软件工作大部分的精力都投入在了需求分析、设计、编程 3 个阶段的开发，造成了这些方面方法论的快速发展，而忽视了测试工作。

总之，与一些发达国家相比，国内软件测试工作还存在一定的差距。主要体现在软件测试意识以及测试理论的研究，大型测试工具软件的开发以及从业人员数量等方面。其实，这与中国整体软件的发展水平是一致的，因为我国整体的软件产业水平和软件发达国家的水平相比有较大的差距，而作为软件产业重要一环的软件测试，必然也存在着不小的差距。但是，我们在软件测试实现方面并不比国外差，国际上优秀的测试工具，我们基本都有，这些工具所体现的思想我们也有深刻的理解，很多大型系统在国内都得到了很好的测试。

1.2 软件测试的基本理论

软件测试在软件生命周期中横跨两个阶段。通常在编写出每个模块之后就对它进行必要的测试(称为单元测试)，模块的编写者和测试者是同一个人，编码和单元测试属于软件生命周期的同一个阶段。在结束这个阶段之后，对软件系统还要进行各种综合测试，这是软件生命周期中另一个独立的阶段，通常由专门的测试人员来完成这项工作。

目前，人们越来越重视软件测试，软件测试的工作量往往占到软件开发总工作量的 40％以上。在特殊情况下，测试那些重大的软件，例如核反应堆监控软件，其测试费用可能相当于软件工程其他步骤总成本的三倍到五倍。因此，我们必须高度重视软件测试工作，绝不能认为写出程序代码之后软件开发工作就完成了。

1.2.1 软件测试的定义和目标

1. 软件测试的定义

人们对于软件测试的目的可能会存在着这样的认识：软件测试是为了证明程序是正

确的。实际上，这种认识是错误的。因为如果表明程序是正确的而进行测试，就会设计一些不易暴露错误的测试方案，也不会主动去检测、排除程序中可能存在的一些隐患。显然，这样的测试对于发现程序中的错误，完善和提高软件的质量作用不大。因为程序在实际运行中会遇到各种各样的实际问题，而这些问题可能是我们在设计时没有考虑到的，所以在设计测试方案时，就应该尽量让它能发现程序中的错误，从而在软件投入运行之前就将这些错误改正，最终把一个高质量的软件系统交给用户使用。

通常对软件测试的定义有如下描述：软件测试是为了发现程序中的错误而执行程序的过程。具体说，它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出一批测试用例，并利用测试用例来运行程序，以发现程序错误的过程。

正确认识测试的目的是十分必要的，只有这样，才能设计出最能暴露错误的测试方案。此外，应该认识到：测试只能证明程序中错误的存在，但不能证明程序中没有错误。因为即使经过了最严格的测试之后，仍然可能还有没被发现的错误存在于程序中，所以说测试只能查出程序中的错误，但不能证明程序没有错误。

2. 软件测试的目标

软件测试工作是非常必要的，测试的目的就在于在软件投入运行之前，尽可能多地发现软件中的错误。软件测试是对软件规格说明、设计和编码的最后复审，是软件质量保证的关键步骤。

实现这个目的关键是如何合理地设计测试用例，在设计测试用例时，要着重考虑那些易于发现程序错误的方法策略与具体数据。

综上所述，软件测试的目的包括以下三点：

(1) 测试是程序的执行过程，目的在于发现错误，不能证明程序的正确性，仅限于处理有限种的情况。

(2) 检查系统是否满足需求，这也是测试的期望目标。

(3) 一个好的测试用例在于发现还未曾发现的错误；成功的测试是发现了错误的测试。

1.2.2　软件测试的标准

软件测试的标准是站在用户的角度，对产品进行全面测试，尽早、尽可能多地发现缺陷(bug)，并负责跟踪和分析产品的问题，对不足之处提出质疑和改进意见。

软件测试标准如下：

(1) 软件测试的目标在于揭示错误。测试人员要始终站在用户的角度去看问题，系统中最严重的错误的是那些导致程序无法满足用户需求的错误。

(2) 软件测试必须基于“质量第一”的思想去开展各项工作。

(3) 事先定义好产品的质量标准。只有建立了质量标准，才能根据测试的结果，对产

品的质量进行分析和评估。

(4) 软件项目一启动,软件测试也就开始了,而不是等程序写完,才开始进行测试。

(5) 测试用例是设计出来的,不是写出来的,所以要根据测试的目的,采用相应的方法去设计测试用例,从而提高测试的效率,更多的发现错误,提高程序的可靠性。

(6) 对发现错误较多的程序段,应进行更深入的测试。

1.2.3 软件测试的原则

各种统计数据显示,软件开发过程中发现缺陷的时间越晚,修复它所花费的成本就越大,因此在需求分析阶段就应当有测试的介入。因为软件测试的对象不仅仅是程序编码,应当对软件开发过程中产生的所有产品都进行测试。这就像造桥梁一样,在图纸上面设计好桥梁的结构之后,我们只有对图纸进行仔细地审查后,才能进行施工。

人们普遍存在着一种观念,认为可以对程序进行完全的测试。如:

- 许多管理者认为存在完全测试的可能性,因此要求员工这样做,并在彼此间确认正在这样做。
- 某些软件测试公司在产品销售说明中保证他们能对软件进行完全的测试。
- 有时,测试覆盖率分析人员为了推销自己,也宣称自己能够分析是否已经对代码进行了完全测试;或者能够指出下一步还需要做什么测试就能够进行完全的测试。
- 许多销售人员向客户强调他们的软件产品经过了完全的测试,彻底没有错误。
- 一些测试人员也相信存在着完全测试的秘诀,甚至为实现这种想法而吃尽了苦头,忍受了数次失败和挫折。但实际上,无论工作得多么辛苦,计划得多么周密,投入的时间多长,人力和物力资源多大,仍然无法做到充分的测试,仍然会遗漏缺陷。

对一个程序进行完全测试就意味着在测试结束之后,再也不会发现其他的软件错误了。其实,这是不可能的,充其量是测试人员的一种美好的愿望而已。

除了测试人员之外,程序员在编写完每段编码之后,或者在每个子模块完成后,都要进行认真测试,这样就可以在最早的时间发现一些潜在的问题并加以解决。之所以这样做,是由于在测试过程中我们要避免一些人为的和主观因素的干扰。我们知道,开发和测试是互为相反的行为过程,两者有着本质的不同。在程序员完成大量的设计和编码之后,让他否定自己所做的工作,是非常不易的,可以说很少有人能有这样的心态。另外一个原因就是系统需求的错误不易被发现,如果程序员检查自己的代码,那么他对系统需求的理解缺乏客观性,往往存在着对问题叙述或说明的误解,不难想象带有错误认识的程序员是很难发现自己程序存在的问题的。

软件测试的本质就是针对要测试的内容确定一组测试用例。测试用例至少应当包括如下几个基本信息: