高等院校软件工程专业规划教材

软件测试技术

免费提供

电子教案

李 凡 田文洪 王伟东 编著

机械工业出版社
CHINA MACHINE PRESS

高等院校软件工程专业规划教材

软件测试技术

李　凡　田文洪　王伟东　编著

机械工业出版社

本书依据软件测试领域的最新国际标准 ISO/IEC/IEEE 29119（2013）系列，系统地介绍了软件测试的基本概念与基本原理，主要的测试用例设计技术，软件生命周期中测试的实施，软件测试的计划、组织与管理，以及软件测试领域中的一些技术性和专业性较强的主题。

本书力求内容精炼、叙述清楚、循序渐进。在对理论和技术细节的阐述中设计了大量实例及步骤讲解，以期读者能够快速、全面地掌握软件测试的相关理论知识、测试用例设计技术，以及软件测试的实施和管理方法，并能够在工程实践中灵活运用。本书每章均配有习题，以指导读者深入地进行学习。

本书既可以作为高等学校软件工程专业、计算机应用专业以及其他相关专业软件测试技术课程的教材，也可以作为软件测试从业人员的技术参考书。

本书配套授课电子课件，需要的教师可登录 www.cmpedu.com 免费注册，审核通过后下载，或联系编辑索取（QQ：2850823885，电话：010-88379739）。

图书在版编目（CIP）数据

软件测试技术 / 李凡，田文洪，王伟东编著. —北京：机械工业出版社，2016.2

高等院校软件工程专业规划教材

ISBN 978-7-111-52680-3

Ⅰ. ①软… Ⅱ. ①李… ②田… ③王… Ⅲ. ①软件－测试－高等学校－教材 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字（2016）第 006591 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：郝建伟 责任校对：张艳霞

责任印制：李 洋

三河市宏达印刷有限公司印刷

2016 年 3 月第 1 版 • 第 1 次印刷

184mm×260mm • 13.25 印张 • 324 千字

0001－3000 册

标准书号：ISBN 978-7-111-52680-3

定价：36.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

服务咨询热线：（010）88379833

读者购书热线：（010）88379649

网络服务

机 工 官 网：www.cmpbook.com

机 工 官 博：weibo.com/cmp1952

教育服务网：www.cmpedu.com

金 书 网：www.golden-book.com

出 版 说 明

计算机软件产业是发展迅速的朝阳产业，软件企业需要大量软件开发的专门技术人才。为了适应社会对软件人才的大量需求，许多高等院校相继开设了软件工程专业以培养专门的软件开发人才。软件工程的主要目标是开发系统模型，研究在有限预算下按时开发高质量软件的可靠技术。软件工程寻找计算机科学中科学与工程的结合，有效地开发和管理软件系统。在人才培养中，教学目标、教学计划、课程建设和教材建设异常重要。为了配合软件工程领域的教材建设，机械工业出版社邀请了高等学校从事软件工程的教学和研究的专家和教师，共同策划了适用于高校软件工程专业的系列教材，主要包括软件工程导论、需求工程、软件体系结构、软件测试技术、软件项目管理、UML 及应用等。

本系列教材的主要特色如下：

1．从培养软件人才的需要出发，规划本系列教材。

2．在内容的选取上，结合了软件工程的最新发展和实际应用，具有先进性和实用性。

3．在教材的编写中，注重理论与实践的结合，注重引入案例，有助于实际能力的培养。

4．在教材的体例上，各章内容具有一定的独立性，便于选择性学习。

本系列教材可作为软件工程专业、计算机科学与技术专业软件工程方向的教学用书，也可作为从事软件开发和软件工程领域工作的科技人员的参考书。

机械工业出版社

前　言

近 30 年来，随着编程语言和软件工程相关技术的快速发展，软件的规模和复杂性大幅度提升，软件的质量问题也变得日益突出。软件中存在的缺陷轻则给用户带来不便，重则造成生命和财产的重大损失。作为软件质量保证的关键步骤，软件测试的效果直接关系到软件产品的质量，因此得到业界的高度重视，软件测试的理论知识和技术工具的发展日新月异。

目前，国内外软件企业或部门对软件测试人员的需求量非常大，掌握软件测试的基本原理、基本方法和基本技术业已成为软件工程及相关专业毕业生的基本要求之一。许多高等院校已经把软件测试技术列为相关专业的必修或选修课程。本书结合编者多年从事软件测试技术相关课程教学的经验，依据软件测试领域的最新国际标准 ISO/IEC/IEEE 29119（2013）系列，系统地介绍了当前软件测试领域的专业知识。在各章节中设计了大量的实例和步骤讲解，尽量做到内容精炼、叙述清楚、循序渐进。通过对本书内容的学习，读者能较快地学习到软件测试方面的理论知识，全面掌握测试用例设计技术，并在软件生命周期内得以灵活运用。因此，本书既可以作为软件测试技术课程的教材，也可以作为软件测试从业人员的技术参考书。

本书共 6 章，可分为 3 部分。第 1 部分包括 1～4 章的内容，介绍了软件测试的概念、原理、方法，以及软件测试的实施等基础内容。其中，第 1 章主要介绍了软件缺陷和软件测试的相关基本概念与基本原理。第 2 章介绍了黑盒测试技术，包括静态黑盒测试技术与动态黑盒测试技术。前者主要包括软件产品规格说明书的审查，后者主要包括等价类划分方法、组合测试设计方法（包括完全组合测试方法、成对测试方法、逐个选择测试方法及基本选择测试方法）、分类树方法、边界值分析方法、决策表方法、因果图方法，以及状态转换测试方法。第 3 章介绍了白盒测试技术，包括静态白盒测试技术与动态白盒测试技术。前者主要包括正式审查方面的内容，后者主要包括语句测试方法、分支测试方法与决策测试方法、分支条件测试方法、分支条件组合测试方法、MCDC 测试方法，以及数据流测试方法。第 4 章介绍了软件生命周期中测试的实施，主要包括递增式测试策略，以及单元测试、集成测试、系统测试、验收测试和回归测试方面的内容，此外还介绍了软件测试实施中常用工具的使用方法。第 2 部分包括第 5 章的内容，主要介绍了软件测试的计划、组织与管理。第 3 部分包括第 6 章的内容，主要介绍了软件测试领域中一些技术性和专业性较强的主题，包括嵌入式软件测试、面向对象软件测试、安全性测试，以及程序变异测试技术。

本书第 1～3 章由李凡编写，第 4～6 章由王伟东、田文洪和李凡共同编写，全书由李凡统稿。本书在编写过程中得到了电子科技大学信息与软件工程学院相关领导的大力支持和机械工业出版社的积极协助，在此表示衷心的感谢。

本书在编写过程中难免存在疏漏和不妥之处，恳请广大专家和读者批评指正。本书编者的电子信箱是 lifan@uestc.edu.cn。

编　者

目　录

第1章　软件测试概述

当前，软件在信息科技（Information Technology，IT）领域中扮演着越来越重要的角色。继苹果公司利用软件重新定义了智能手机和平板电脑，并引发了行业的革命性变化之后，软件定义的网络（Software-defined Networking，SDN）、软件定义的数据中心（Software-defined Data Center，SDDC）及软件定义的存储（Software-defined Storage，SDS）等思想、概念和产品不断涌现。除了IT产品外，软件也日益渗透到人们的日常生活中，电视机、电冰箱、手表和汽车等传统工业产品也已经或即将加入被软件所定义的行列（如谷歌眼镜、无人驾驶汽车等）。简而言之，软件正在重新定义一切（Software-defined Anything，SDX）。

在这种大背景之下，由于软件缺陷而造成重大经济损失乃至人员伤亡等严重后果的事例屡见不鲜，因此，如何保证软件产品的质量就成为必须解决的一个关键问题。

软件测试是软件质量保证的关键步骤，也是软件工程的重要组成部分，对软件产品的质量起着至关重要的作用。本章概述了与软件测试相关的基本概念，为读者建立起对软件测试的全面了解，并为进一步讨论软件测试技术奠定基础。

1.1　软件与软件缺陷

软件测试的对象——软件，是一种特殊的逻辑实体，其特点导致了产生软件缺陷几乎是不可避免的。软件测试工作的目的就在于尽早找到软件缺陷，并保证其得到修复。在讨论软件测试之前，应该对软件与软件缺陷的相关基本概念和基本知识进行讨论。本节将介绍这方面的内容。

1.1.1　软件及其特点

IEEE Std 610.12 给出了软件（Software）的定义：软件是计算机程序及其相关的文档，以及用以操作计算机系统的附属数据。

一般而言，软件来源于应用问题。应用问题经过抽象（包括数据抽象和过程抽象）后，形成软件的设计（包括数据结构和算法等内容），继而被编码实现。其后软件即可运行于硬件之上。通常，软件的运行还包括操作系统的支持。上述过程如图1-1所示。

从软件的定义可以看出，软件有3个主要组成部分。

1）指令的集合（计算机程序），通过执行这些指令可以满足预期的特征、功能和性能需求。

2）数据结构，它使得程序可以利用必要的信息。

3）描述程序操作和使用的文档。

其中，指令集合和数据结构是软件的可运行部分，而文档是软件的不可运行部分。

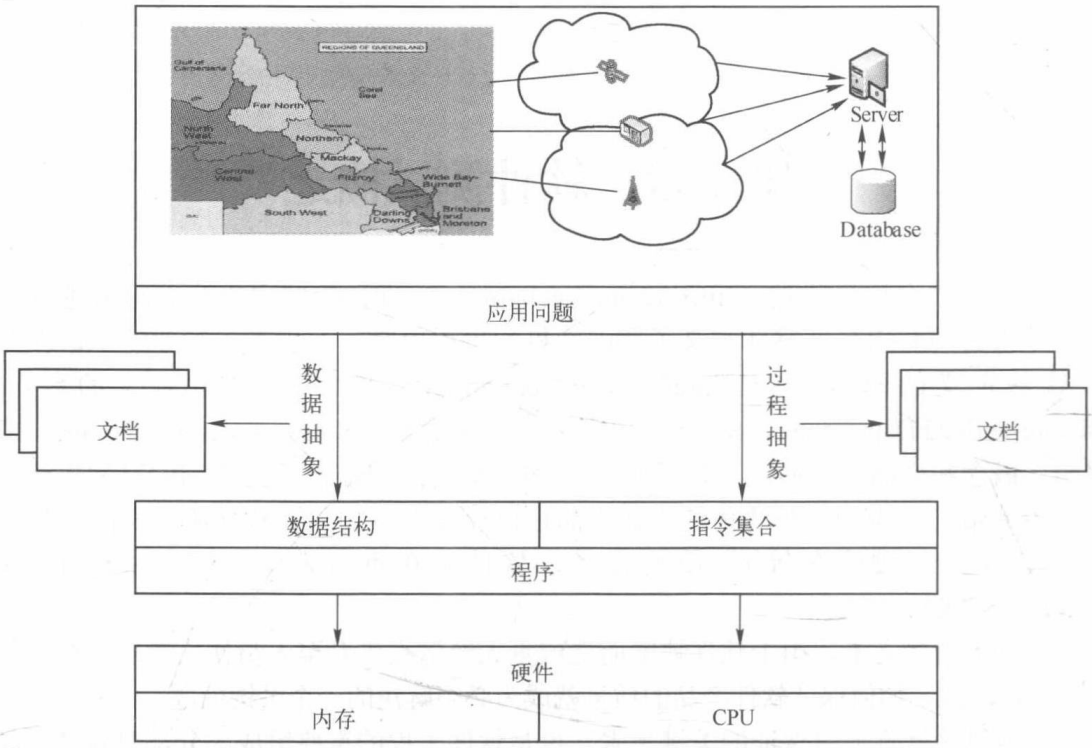

图 1-1　软件的产生与运行

软件的产生和运行方式决定了软件具有以下特点。

1）软件是一种逻辑实体，而不是具体的物理实体，因而它具有抽象性。

2）软件是设计和开发出的，与硬件不同，在其开发过程中没有明显的制造过程。

3）在软件的运行和使用期间，没有硬件那样的机械磨损、老化等问题。

4）虽然软件工业向着基于构件的构造模式发展，但目前大多数软件仍是依据客户的需求定制的。

1.1.2　软件缺陷

软件的特点决定了软件中存在缺陷是难以避免的。在对软件缺陷进行具体分析前，首先对软件缺陷进行明确的定义。

1. 软件缺陷的定义

IEEE Std 729 中定义了软件缺陷，具体如下。

从产品内部看，软件缺陷是软件产品开发或维护过程中所存在的错误、毛病等各种问题。从产品外部看，软件缺陷是系统所需要实现的某种功能的失效或违背。

简而言之，软件缺陷就是软件产品中所存在的各种问题，其最终表现为没有满足用户的需求。

由软件缺陷的定义，可以得出软件缺陷的主要表现形式，具体如下。

1）软件未实现产品规格说明书要求的功能。

2）软件出现了产品规格说明书中指明不会出现的错误。

3）软件实现了产品规格说明书中未提到的功能[①]。

① 见本章“习题与练习”部分第 1 大题的第 3）题。

4）软件未实现产品规格说明书虽未明确提及但应该实现的目标。

5）软件难以理解、不易使用、运行缓慢，或者从测试人员的角度看，最终用户会认为不好。

2. 软件缺陷的主要来源

软件的设计和开发目前尚未完全摆脱手工开发方式，“人”的因素难以避免，因此从设计和开发的角度看，软件的特点决定了软件缺陷是软件产品的固有属性。

首先，由于软件是逻辑实体，因此其可视性低，缺乏直观性，在开发上存在困难。其次，有一定规模的软件，其复杂程度是非常高的，例如软件的输入空间很大，可执行路径非常多。这些复杂性通常是开发人员难以理解和把握的。第三，大多数软件都是定制的，受到充分测试的可选标准构件的使用有很大的局限性。实证研究表明，每写 1000 行代码会产生 30～85 个缺陷。此外，软件的开发和运行对计算机系统有着不同程度的依赖性，常常会受到计算机系统的限制，因此软件中不出现软件缺陷几乎是不可能的事情。

一般而言，软件缺陷的主要来源有以下几点。

（1）软件缺陷的首要来源是软件产品规格说明书

软件产品规格说明书是需求分析阶段得出的最主要的文档，要求完整、准确、具体地描述系统的数据要求、功能需求、性能需求、可靠性和可用性要求、出错处理需求、接口需求、约束、逆向需求，以及将来可能提出的要求。

虽然已经有若干以形式化方法描述用户对软件系统的需求的研究结果，但目前大多数软件需求分析的结果仍然是非形式化或半形式化的，即以自然语言描述，或以自然语言描述为主，辅之以数据流图、实体-联系图（E-R 图）等建立模型。用非形式化或半形式化描述，难以避免软件产品规格说明书中出现不一致、歧义、含糊、不完整及抽象层次混乱等问题，这些问题映射到软件的设计与实现当中，将很可能导致软件缺陷的出现。

另外一种常见的情况是，在做软件需求分析时，并没有充分理解用户的需求，或者对用户需求的理解产生了偏差，导致产品规格说明书出现问题。另外，在软件开发过程中，用户需求发生变化，导致修改产品规格说明书时也通常会引入大量问题。

（2）软件缺陷的第二大来源是设计

在软件的概要设计阶段，应确定系统的物理配置方案，进而确定软件的结构，包括模块的划分、调用关系等内容。在软件的详细设计阶段，应确定怎样具体实现用户需要的软件系统，也就是要设计出程序的“蓝图”。无论是概要设计阶段还是详细设计阶段，设计中都可能发生错误，同时在设计过程中也存在着人的因素，经常导致随意、易变和沟通不足等问题。以上因素都可能导致软件的设计出现问题，继而导致缺陷的产生。

（3）软件缺陷的第三大来源是编码错误

编码错误主要由于软件的复杂性、文档不足、进度压力或者普通的技术性错误。这些在软件开发过程中几乎都是难以避免的。

以软件开发中常用的二分搜索算法为例，历史上第一个二分搜索算法早在 1946 年就出现了，但是第一个完全正确的算法直到 1962 年才出现。Bentley 曾指出，90%的计算机专家不能在两小时内写出完全正确的二分搜索算法。由此可见，软件的编码错误是普遍存在的。

又如必应词典（运行界面见图 1-2），这款软件的主要功能是将用户输入的英文单词发送到服务器，然后将服务器返回的中文解释显示在界面上。就功能而言，该软件相对并不复杂，即使这样，该软件也拥有 40 多个线程、1000 多个 Windows 句柄，加载 170 多个动态链接库

（具体见参考文献[4]）。可以说，几乎没有一个开发人员能完全理解它所用的所有技术，也无法掌握软件在任何时刻的状态。

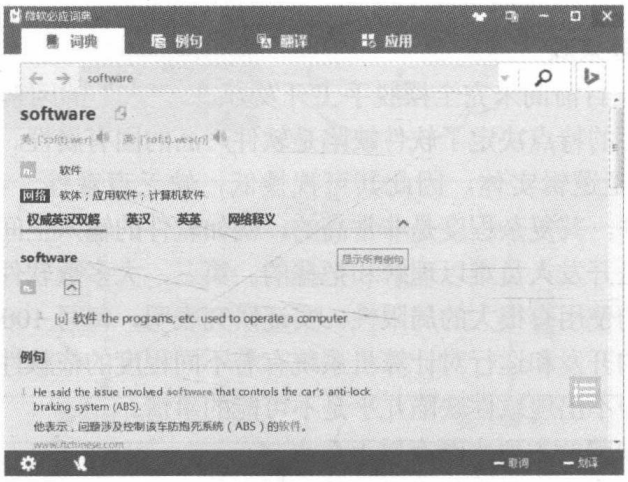

图 1-2　微软必应词典运行界面

为了使得开发人员能从繁杂的技术细节中摆脱出来，出现了高抽象层次的编程语言、程序库及框架等软件开发技术。这些技术无疑能极大程度地提高开发者的开发效率，但使用这些技术却导致了开发人员与代码底层细节的割裂。在具体软件开发中总存在着一些情况，需要开发人员深入理解代码的底层细节，因此现代软件开发技术的使用虽然减少了代码的编写时间，但却可能引入大量的隐蔽性极强的软件缺陷。

还必须指出的是，许多看上去是编程错误的软件缺陷实际是由产品规格说明书和软件设计方案造成的，而并非单纯的编程错误。

（4）其他原因

其他原因主要包括误解、重复错误和测试错误等，它们只占极小的比例。

以上软件缺陷的主要来源的大致占比如图 1-3 所示。

图 1-3　软件缺陷的主要来源

3. 软件缺陷的相关术语

软件缺陷有如下几个相关术语。

（1）软件失效（Failure）。

软件失效是指不完全符合给定的需求，是实际结果或行为（执行测试时观察到的）与期望结果或行为（规格说明或需求中定义的）之间的偏差。IEEE Std 610.12 使用该术语描述用户遇到问题的情形。

（2）软件故障（Fault）

软件故障也称为缺点（Defect）、缺陷（Bug），是指软件中的静态缺陷。软件的每个故障都是软件开发或更改后就存在的，它是设计错误，不是自发出现的。只有在软件执行过程中，这些故障才以失效的方式显露出来。

（3）软件错误（Error）

软件错误是指不正确的内部状态，该状态是某个故障的表现。

以上 3 个术语分别描述了软件缺陷的不同层次。软件故障一般源自于人为的失误（也有

可能是硬件问题），例如编程错误。在执行软件的过程中，软件故障引发软件内部状态与预期不一致，即产生了软件错误。最终从输出的角度看，软件没有符合给定的需求，即软件的行为表现出软件失效。

软件错误在很多情况下会导致软件失效，但软件的错误并不一定会引发软件失效。参考下面的 Java 程序。

【例 1-1】 软件错误、软件故障与软件失效。

```
1    public static int lastZero (int[] x){
2        //返回最后一个非 0 元素的数组下标，若 x 中没有 0 元素则返回-1
3        for (int i=0; i < x.length; i++){
4            if (x[i] == 0)
5                return i;
6        }
7        return -1;
8    }
```

解：该段代码寻找的不是最后一个非 0 元素的数组下标，而是第一个非 0 元素的数组下标。for (int i=0; i < x.length; i++)一句应为 for (int i=x.length-1; i >= 0; i--)。因此这段代码中存在着故障。考虑输入 x={1,0,3}时的情况。程序中 for 循环的第一次迭代中，在执行到第 4 行的 if 语句时，程序的状态为：i=0，x={1,0,3}，PC=if[①]。此状态是错误的，因为 i 应该为 2 而非 0。因此程序的故障导致了程序的错误。但是在 x={1,0,3}下，程序的输出为 1，恰好为预期输出结果。因此程序的错误状态并没有传播到输出，即此时软件错误没有引发软件失效。

同理，软件故障也不一定会引发软件错误。例如在【例 1-1】中，当 x={0}时，软件的故障并没有引发软件错误，因此也不会导致软件失效。但当 x={3}时，软件的故障就可以引发软件错误，请读者自行思考其中的原因。

还有一种更为复杂的情况，某个故障可能会被软件的其他部分的某个或某些故障所掩盖，这种情况称为故障屏蔽（Defect Masking）。在这种情况下，只有修正了屏蔽它的故障后，相应的失效才会显现出来。

4. 软件缺陷的等级

虽然从外部看，软件缺陷都是系统所需要实现的某种功能的失效或违背，但不同的软件缺陷对用户使用的影响是不同的，所造成的后果也是不同的。例如界面上的小问题与用户数据丢失显然是不同的。另一方面，如果项目有充足的时间，上述两个问题都应该解决。但由于时间和经费的限制，在每一个软件项目中都必须进行取舍，以决定哪些软件缺陷应修复，哪些不应修复，哪些推迟到软件的以后版本中解决。

因此，软件缺陷并不能平等对待。必须对软件缺陷进行分类，软件测试员才能以简明扼要的方式指出其影响。可以从上述两个出发点对软件缺陷划分严重性（Severity）和优先级（Priority）。

（1）严重性

严重性是指对用户的影响程度。一般可以定义下面 4 种级别。

① 此处 PC 指程序计数器。

1）致命（Fatal），指产生系统崩溃、数据丢失、数据毁坏和安全性破坏等非常严重的情况。

2）严重（Critical），指功能或特性没有实现、操作性错误及结果错误等情况。

3）一般（Major），指没有产生严重的结果，但影响外观或用户使用，如出现拼写错误、UI 布局不良等小问题。

4）微小（Minor），指建议性的问题。

（2）优先级

优先级是指缺陷修复的紧急程度。一般可以定义下面 4 种级别。

1）马上解决，指明显的问题，阻碍了进一步测试。

2）高优先级，指缺陷严重，必须在产品发布之前修复。

3）普通优先级，指正常排队等待修复，如果时间允许应该予以修复。

4）低优先级，指可能会修复，但是即使存在该级别的缺陷，产品也能发布。

严重性和优先级的各个级别通常情况下是对应的，但也有例外情况。例如，软件产品启动画面上公司的名称拼写错误，就是低严重性但高优先级的缺陷。

严重性和优先级对于审查缺陷报告并决定哪些软件缺陷应该修复，以何种顺序修复是很重要的（详见第 5 章）。值得注意的是，软件缺陷的优先级在其生命周期中可能会发生变化。测试员需要持续跟踪缺陷的状态，确保缺陷在优先级发生变化时能及时发现并处理。

5. 软件缺陷的生命周期

软件缺陷的生命周期指的是软件缺陷从被发现到最后被关闭的整个过程。最简单也是最优化的软件缺陷生命周期如图 1-4a 所示。

测试人员发现缺陷后，缺陷被记录下来并分配给指定的程序员修复，此时缺陷处于打开状态。开发人员修复缺陷后，提交报告给测试人员，此时缺陷处于解决状态。最后测试人员确认缺陷得到修复后，将其关闭，此时缺陷处于关闭状态。

但在实际工作中，软件缺陷经常经历数次改动和状态变化，有时可能重新开始其生命周期，如图 1-4b 所示。

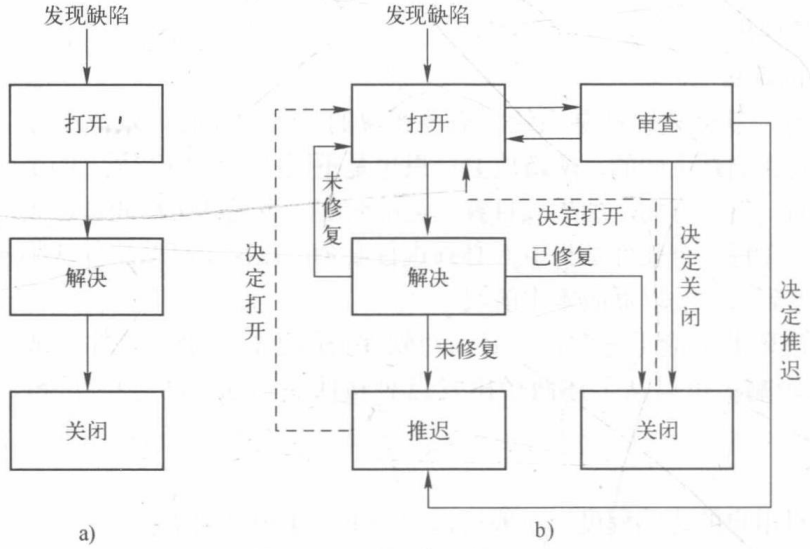

图 1-4　软件缺陷的生命周期

a) 最简化的软件缺陷生命周期　b) 实际工作中的软件缺陷生命周期

图 1-4b 中的审查状态是指项目经理或管理小组决定软件缺陷是否修复。如果决定不予修复，则直接进入关闭状态。推迟状态是指决定软件缺陷不马上修复，而是留到将来考虑。如果在解决状态下，测试人员发现缺陷没有得到修复，则返回打开状态，开始新的生命周期。在解决状态下，测试人员发现缺陷没有得到修复，也可能进入推迟状态，留到将来解决。从关闭状态和推迟状态回到打开状态的情况发生较少，因而用虚线表示。通常在以下情况下发生。

1）原来认为已经修复而关闭的缺陷在测试中再次发生。

2）原来认为可以推迟解决的缺陷被发现严重性上升，必须马上得到修复。

从软件缺陷生命周期的描述中不难发现，在整个软件缺陷的生命周期中，测试人员、开发人员及管理人员必须紧密配合，及时沟通，才能达到提高软件质量、降低成本的目的。

1.1.3 软件缺陷的典型案例

目前，软件已经深入渗透到人类生活的各个方面。软件在生命周期的各个阶段都有可能发生软件缺陷，其表现形式和后果也各不相同，有些只像是一个玩笑，但在一些情况下，会导致开发商和用户的成本大幅上升，在某些安全性至关重要的场合下，甚至还可能导致人员伤亡的严重后果。

1. 火星气候探测者故障

火星气候探测者（Mars Climate Orbiter）是美国太空总署的火星探测卫星，也是火星探测计划的一部分，于 1999 年发射。火星气候探测者号在进入火星轨道的过程中失去联络，任务失败。火星气候探测者号按原计划应进入距火星 140～150 千米的轨道上。但是一个导航错误致使该飞船接近至火星 57 千米处。飞船因此被这一高度上的大气压力和摩擦所摧毁。

该导航错误产生的原因是参与该项目的洛克希德.马丁公司使用了英制单位的加速度数据（磅-秒制），而另一参与该项目的喷气推进实验室采用公制（牛顿-秒制）加速度进行计算，该错误导致卫星入轨不正常。但卫星在进入异常状态时，系统的异常处理机制却未被触发。

2. Therac-25 放射治疗仪事件

Therac-25 是一种由加拿大原子能有限公司（Atomic Energy of Canada Limited，AECL）和法国 CGR 公司联合开发的放射疗法治疗仪。在 1985 年到 1987 年间，该仪器涉及至少 6 起医疗事故。事故中的病人接受了约 100 倍于正常剂量的辐射。6 起事故中有 3 个病人因辐射的直接影响而死亡。

Therac-25 提供两种放射治疗模式：直接电子束治疗和 Megavolt X 射线治疗。设备软件的设备控制任务和用户操作任务没有很好的同步。当操作人员过快地在两种模式之间切换时可能发生竞争条件，使得系统进入不安全状态。虽然软件代码中设有安全防护程序，但由于软件中一个标记变量处理不当，存在溢出问题，使得上述安全防护程序失效。事实上，Therac-25 重用了老型号 T-20 的软件代码，而 T-20 中设有硬件互锁装置防止电子束在某个部件没有就位的情况进入高能状态，使得这种缺陷状态永不会被触发，从而掩盖了软件中的缺陷。而 Therac-25 过于依赖软件控制，省去了上述硬件互锁装置，使得缺陷暴露了出来。

3. Ariane 5 型飞船故障

1996 年 6 月 4 日，Ariane 5 发射 40 秒后爆炸，发射任务失败。任务失败的原因是系统软件将一个 64 位浮点值转换为 16 位有符号整数值时，超出了 16 位整数的表示范围，而这个异常未得到正确处理。该错误产生的根本原因在于设计 Ariane 5 时，做了 Ariane 5 和 Ariane 4 具有相同环境的假设，重用的软件在新的环境下完全没有进行测试。此外，错误处理模块的处理

机制不正确。

4. 英特尔奔腾浮点除法缺陷

在 1994 年 10 月 30 日，Thomas R. Nicely 博士在实验中发现了奔腾 CPU 的除法问题：(4195835/3145727) ×31435727-4195835 的结果不为 0。他把发现的问题放到因特网上，引发了大量的讨论，并且发现在其他一些特定条件下也会出现类似的问题。

事实上，英特尔公司的软件测试工程师在芯片发布前进行的内部测试时就已经发现了这个问题，但该公司的管理层认为这个缺陷没有严重到需要修正和公开的程度。当软件缺陷发现时，该公司又试图弱化该问题的严重性。在受到压力承诺更换有问题的芯片时，该公司又提出要求，让用户自己证明自己受到缺陷的影响。最终该问题越演越烈，导致英特尔公司为软件缺陷的行为公开道歉，并花费 4 亿美元支付更换问题芯片的费用。

5. Windows 蓝屏故障

在 CES 2005 展会第一天，盖茨介绍了微软的“无缝计算”战略。但就在他演示时，Windows 出现了蓝屏，引起观众的哄笑。

1.1.4 软件缺陷的经济影响及修复费用

美国商务部下属的国立标准技术研究所（National Institute of Standards and Technology，NIST）在其发布的报告“The Economic Impacts of Inadequate Infrastructure for Software Testing (2002)”中指出，“据推测，由于软件缺陷而引起的损失额每年高达 595 亿美元。这一数字相当于美国国内生产总值的 0.6%”。可以预见，随着时间的推移，软件缺陷的经济影响将进一步加大。

软件在其生命周期的各个阶段都有可能发生问题，但修复软件缺陷的费用在软件生命周期的各个阶段是有很大差异的。一般的规律是，随着时间推移，修复软件缺陷的费用呈指数级增长。

在这方面，已经有大量的研究成果。例如 IBM 的 Watts Humphrey 指出，确定软件错误的相对成本是：在设计阶段，1.5；编码前，1；编码中，1.5；测试前，10；测试中，60；交付后，100。

软件缺陷的修复费用随时间推移的变化情况如图 1-5 所示。

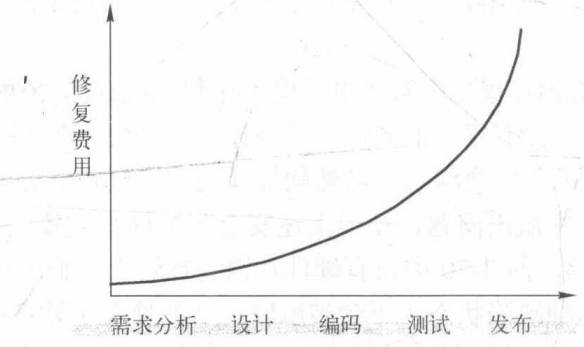

图 1-5 软件缺陷修复费用

1.2 软件测试及其发展

软件测试是软件工程的重要组成部分，也是软件质量保证的关键步骤，用以确认软件的