普通高等应用型院校“十二五”规划教材

软件测试基础教程（第二版）

主　编　杜文洁　王占军　高　芳　副主编　高为民　罗　旭　刘　冰　周　颖

中国水利水电出版社
www.waterpub.com.cn

普通高等应用型院校“十二五”规划教材

软件测试基础教程

（第二版）

主　编　杜文洁　王占军　高　芳

副主编　高为民　罗　旭　刘　冰　周　颖

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

软件测试技术是软件产业发展的重要因素，它对保障软件产品质量有着举足轻重的作用。本书详尽地阐述了软件测试基础知识及其相关的实用技术，内容包括软件测试的基础理论、软件测试方法、软件测试流程、软件测试环境的搭建、黑盒测试实例设计、白盒测试实例设计、软件测试计划与文档、软件自动化测试、面向对象的软件测试、Web 网站测试和云计算对软件测试的影响。本书结合教学实例突出基本知识和基本概念的表述，注重内容的先进性、系统性和实用性，力求反映软件测试发展的最新成果。将测试与软件工程密切结合，使读者可以更好地理解和掌握软件测试的内容，并迅速地运用到实际测试工作中去。

本书可作为高等院校计算机相关专业的软件测试课程教材，也可作为软件测试技术学习和提高的培训教材，亦可供从事软件开发和软件测试工作的技术人员参阅。

本书配有电子教案，读者可以到中国水利水电出版社网站和万水书苑上免费下载，网址为 http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

软件测试基础教程 / 杜文洁，王占军，高芳主编
-- 2版. -- 北京 : 中国水利水电出版社，2016.1
普通高等应用型院校“十二五”规划教材
ISBN 978-7-5170-3972-3

Ⅰ. ①软… Ⅱ. ①杜… ②王… ③高… 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(2015)第321330号

策划编辑：石永峰　责任编辑：张玉玲　加工编辑：孙　丹　封面设计：李　佳

书　名	普通高等应用型院校“十二五”规划教材 软件测试基础教程（第二版）
作　者	主　编　杜文洁　王占军　高　芳 副主编　高为民　罗　旭　刘　冰　周　颖
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（发行部）、82562819（万水）
经　售	北京科水图书销售中心（零售） 电话：（010）88383994、63202643、68545874 全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	三河市铭浩彩色印装有限公司
规　格	184mm×260mm　16 开本　13.25 印张　324 千字
版　次	2008 年 3 月第 1 版　2008 年 3 月第 1 次印刷 2016 年 1 月第 2 版　2016 年 1 月第 1 次印刷
印　数	0001—3000 册
定　价	27.00 元

第二版前言

本书第一版出版以来，读者反映效果良好。近年来，国内的软件测试技术日益完善和成熟，就业市场对高水平的软件测试人才需求量日益增大。为了进一步深化软件测试课程的教学改革，培养高质量的软件测试人员，在听取行业专家和读者意见的基础上，结合各高等院校软件测试课程的实际教学情况，编写了软件测试基础教程（第二版）。

本书第二版主要基于第一版内容的基础上，增加了一些内容。在结构安排上遵循系统化和简明化原则，由浅入深、层层推进，达到好教易学的效果。在语言表述上注重理论通俗易懂，例子形象实用，使学生将理论知识与实际应用充分结合。

本书共分 11 章，系统地介绍了软件测试的主要内容，具体分布如下：

第 1 章　软件测试的基础理论。介绍了软件测试的相关理论、生命周期，以及软件测试与软件开发的关系。

第 2 章　软件测试方法。概括介绍了软件测试的相关方法，具体介绍了两组测试方法，分别是静态测试与动态测试、黑盒测试与白盒测试。

第 3 章　软件测试流程。介绍了软件测试的复杂性与经济性分析。描述了软件测试的流程和策略，其中包括单元测试、集成测试、确认测试、系统测试和验收测试 5 个测试阶段。

第 4 章　软件测试环境的搭建。介绍了测试环境的作用、要素，描述了如何搭建测试实验室及其日常管理和维护。

第 5 章　黑盒测试实例设计。介绍了等价类划分法，边界值分析法，决策表法，因果图法以及黑盒测试综合用例。

第 6 章　白盒测试实例设计。介绍了逻辑覆盖测试，路径分析测试，其他白盒测试方法以及白盒测试综合用例。

第 7 章　软件测试计划与文档。详细阐述了测试计划的制定、测试文档的主要内容和软件生存周期各阶段的测试任务与可交付的文档，列举了测试用例、测试总结报告的设计内容。

第 8 章　软件自动化测试。介绍了软件自动化测试、自动化测试的设计与开发以及常用的自动化测试工具。

第 9 章　面向对象的软件测试。主要介绍了面向对象测试与传统测试的区别、面向对象的测试方法以及类测试。

第 10 章　Web 网站测试。介绍了 Web 网站的测试、功能测试、性能测试、安全性测试、导航测试、配置和兼容性测试以及数据库测试。

第 11 章　云计算对软件测试的影响。介绍了云计算及云测试的概念；分析了云测试的优势；阐述了云计算与云测试的发展对软件测试发展的影响。

本书由杜文洁、王占军、高芳任主编，高为民、罗旭、刘冰、周颖任副主编，另外，周功、杨柠、李虹等参与了部分内容的编写。全书由杜文洁统稿完成。

由于作者水平和时间有限，书中难免出现一些疏漏，请读者批评指教。

作　者

2015 年 10 月

目　　录

第 1 章　软件测试的基础理论

本章概述

介绍了软件测试的发展历史及其现状，软件测试的定义、测试目的、测试原则、测试的生命周期，阐述了软件测试与软件开发的关系。

1.1　软件测试的含义

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。

在开发软件的过程中，人们使用了许多保证软件质量的方法分析、设计和实现软件，但难免还会在工作中犯错误。这样，在软件产品中就会隐藏许多错误和缺陷。对于规模大、复杂性高的软件更是如此。在这些错误中，有些是致命的错误，如果不排除，就会导致生命与财产的重大损失。

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。如何防止和减少这些可能存在的问题呢？那就是进行软件测试。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。

正如食品生产厂家在把产品销售给商家之前要进行合格检验一样，软件企业在把软件提交给客户之前也需要进行严格的测试。如果把所开发出来的软件看作一个企业生产的产品，那么软件测试就相当于该企业的质量检测部分。简单地说，我们在编写完一段代码之后，检查其是否如我们所预期的那样运行，这个活动就可以看作是一种软件测试工作。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. 软件缺陷案例

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这个市场，所以进行了大量促销宣传。结果，销售额非常可观，该游戏成为孩子们那年节假日的“必买游戏”。然而后来却飞来横祸。12 月 26 日，圣诞节的后一天，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。因为工资系统相当信赖于日期的处理，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 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。最令人警醒的问题是，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地数据是否置位——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33=9的错误；又如用鼠标取词“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。如果得出其他结果，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，弗吉利亚州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的留的反馈意见。

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。软件缺陷（bug），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。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复杂且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、过程改进提供了许多信息。

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）软件超出实现了产品说明书提到的功能。

（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，说明书中从未提到这一功能，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，会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“＝”键布置的位置使其极其不好按；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。

3. 软件缺陷的种类

软件缺陷表现的形式有多种，不仅仅体现在功能的失效方面，还体现在其他方面。软件缺陷的主要类型有：

- 功能、特性没有实现或部分实现。
- 设计不合理，存在缺陷。
- 实际结果和预期结果不一致。
- 运行出错，包括运行中断、系统崩溃、界面混乱。
- 数据结果不正确、精度不够。

用户不能接受的其他问题，如存取时间过长、界面不美观。

4. 软件缺陷的级别及软件缺陷的状态

（1）软件缺陷的级别

作为软件测试员，可能所发现的大多数问题不是那么明显、严重，而是难以觉察的简单而细微的错误，有些是真正的错误，也有些不是。一般来说，问题越严重的，其优先级越高，越要得到及时的纠正。软件公司对缺陷严重性级别的定义不尽相同，但一般可以概括为 4 种级别：

- 致命的：致命的错误，造成系统或应用程序崩溃、死机、系统悬挂，或造成数据丢失、主要功能完全丧失等。
- 严重的：严重错误，指功能或特性没有实现，主要功能部分丧失，次要功能完全丧失，或致命的错误声明。
- 一般的：不太严重的错误，这样的软件缺陷虽然不影响系统的基本使用，但没有很好地实现功能，没有达到预期效果。如次要功能丧失，提示信息不太准确，或用户界面差、操作时间长等。
- 微小的：一些小问题，对功能几乎没有影响，产品及属性仍可使用，如有个别错别字、文字排列不整齐等。

除了这 4 种之外，有时需要“建议”级别来处理测试人员所提出的建议或质疑，如建议程序做适当的修改，来改善程序运行状态，或对设计不合理、不明白的地方提出质疑。

（2）软件缺陷的状态

软件缺陷除了严重性之外，还存在反映软件缺陷处于一种什么样的状态，便于跟踪和管理某个产品的缺陷，可以定义不同的 bug 状态。

- 激活状态：问题还没有解决，测试人员新报的 bug，或验证后 bug 仍然存在。
- 已修正状态：开发人员针对所存在的缺陷修改程序，认为已解决问题，或通过单元测试。
- 关闭或非激活状态：测试人员验证已经修正的 bug 后，确认 bug 不存在以后的状态。

5. 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，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，没有得到充分的沟通。

（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、可靠性方面的隐患。

- 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。

6. 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，结果规格说明书是软件缺陷出现最多的地方，见图 1-1。

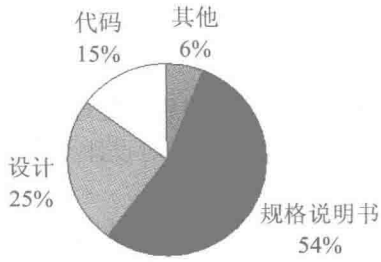

图 1-1　软件缺陷构成示意图

软件产品规格说明书是软件缺陷存在最多的地方，其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，容易引起前后文和上下文的矛盾。

（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。

1.1.2　软件测试技术的发展历史及现状

1. 软件测试技术的发展历史

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，各种高级语言相继诞生，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，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管理、分析、设计、测试和部署等工作，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管理、分析、设计 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，国内测试工作还存在一定的差距。主要体现在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与软件发达国家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 软件测试的目的与原则

1. 软件测试的定义

为了保证软件的质量和可靠性，应力求在分析、设计等各个开发阶段结束前，对软件进行严格的技术评审。但由于人们能力的局限性，审查不能发现所有的错误。而且在编码阶段还会引进大量的错误。这些错误和缺陷如果遗留到软件交付投入运行之时，终将会暴露出来。但到那时，不仅改正这些错误的代价更高，而且往往造成很恶劣的后果。

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。通常对软件测试的定义有如下描述：

软件测试是为了发现错误而执行程序的过程。或者说，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，并利用这些测试用例去运行程序，以发现程序错误的过程。

软件测试在软件生存期中横跨两个阶段：通常在编写出每一个模块之后就对它做必要的测试（称为单元测试）。编码与单元测试属于软件生存期中的同一个阶段。在结束这个阶段之后，对软件系统还要进行各种综合测试，这是软件生存期的另一个独立的阶段，即测试阶段。

现在，软件开发机构将研制力量的 40%以上投入到软件测试之中的事例越来越多。特殊情况下，对于性命攸关的软件，例如飞行控制、核反应堆监控软件等，其测试费用甚至高达所有其他软件工程阶段费用总和的 3～5 倍。

2. 软件测试的目的

基于不同的立场，存在着两种完全不同的测试目的。从用户的角度出发，普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否可以接受该产品。而从软件开发者的角度出发，则希望成为表明软件产品中不存在错误的过程，验证该软件已正确地实现了用户的要求，确立人们对软件质量的信心。因此，他们会选择那些导致程序失效概率小的测试用例，回避那些易于暴露程序错误的测试用例。同时，也不会注意去检测、排除程序中可能包含的副作用。显然，这样的测试对完善和提高软件质量毫无价值。因为在程序中往往存在着许多预料不到的问题，可能会被疏露，许多隐藏的错误只有在特定的环境下才可能暴露出来。如果不把着眼点放在尽可能查找错误这样一个基础上，这些隐藏的错误和缺陷就查不出来，会遗留到运行阶段中去。

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。

3. 软件测试的原则

软件测试的目标是想以最少的时间和人力找出软件中潜在的各种错误和缺陷。如果成功地实施了测试，就能够发现软件中的错误。

根据这样的测试目的，软件测试的原则如下：

（1）应当把尽早地和不断地进行软件测试作为软件开发者的座右铭。坚持在软件开发的各个阶段的技术评审，这样才能在开发过程中尽早发现和预防错误，把出现的错误克服在早期，杜绝某些隐患，提高软件质量。

（2）测试用例应由测试输入数据和与之对应的预期输出结果这两部分组成。如果对测试输入数据没有给出预期的程序输出结果，那么就缺少了检验实测结果的基准，就有可能把一个似是而非的错误结果当成正确结果。

（3）程序员应避免检查自己的程序。如果由他人来测试程序员编写的程序，可能会更客观、更有效，并更容易取得成功。

（4）在设计测试用例时，应当包括合理的输入条件和不合理的输入条件。合理的输入条件是指能验证程序正确的输入条件；而不合理的输入条件是指异常的、临界的、可能引起问题变异的输入条件。因此，软件系统处理非法命令的能力也必须在测试时受到检验。用不合理的输入条件测试程序时，往往比用合理的输入条件进行测试能发现更多的错误。

（5）充分注意测试中的群集现象。测试时不要以为找到了几个错误问题就已解决，不需要继续测试了。应当对错误群集的程序段进行重点测试，以提高测试投资的效益。

（6）严格执行测试计划，排除测试的随意性。对于测试计划，要明确规定，不要随意解释。

（7）应当对每一个测试结果做全面检查。这是一条最明显的原则，但常常被忽视。必须

对预期的输出结果明确定义，对实测的结果仔细分析检查，抓住关键，暴露错误。

（8）妥善保存测试计划，测试用例，出错统计和最终分析报告，为维护提供方便。

4. 软件测试的分类

从不同的角度，可以把软件测试技术分成不同种类。

（1）从是否需要执行被测软件的角度分类

从是否需要执行被测软件的角度，可分为静态测试（Static Testing）和动态测试（Dynamic Testing）。顾名思义，静态测试就是通过对被测程序的静态审查，发现代码中潜在的错误。它一般用人工方式脱机完成，故亦称人工测试或代码评审（Code Review）；也可借助于静态分析器在机器上以自动方式进行检查，但不要求程序本身在机器上运行。按照评审的不同组织形式，代码评审又可分为代码会审、走查、办公桌检查、同行评分四种。对某个具体的程序，通常只使用一种评审方式。

动态测试的对象必须是能够由计算机真正运行的被测试的程序。它分为黑盒测试和白盒测试，也是我们下面将要介绍的内容。

（2）从软件测试用例设计方法的角度分类

从软件测试用例设计方法的角度，可分为黑盒测试（Black-Box Testing）和白盒测试（White-Box Testing）。

黑盒测试是一种从用户观点出发的测试，又称为功能测试，数据驱动测试和基于规格说明的测试。使用这种方法进行测试时，把被测试程序当作一个黑盒，忽略程序内部结构的特性，测试者在只知道该程序输入和输出之间的关系或程序功能的情况下，依靠能够反映这一关系和程序功能需求规格的说明书，来确定测试用例和推断测试结果的正确性。简单地说，若测试用例的设计是基于产品的功能，目的是检查程序各个功能是否实现，并检查其中的功能错误，则这种测试方法称为黑盒。

白盒测试基于产品的内部结构来进行测试，检查内部操作是否按规定执行，软件各个部分功能是否得到充分利用。白盒测试又称为结构测试、逻辑驱动测试或基于程序的测试。即根据被测程序的内部结构设计测试用例，测试者需事先了解被测试程序的结构。

（3）从软件测试的策略和过程的角度分类。

按照软件测试的策略和过程分类，软件测试可分为单元测试（Unit Testing）、集成测试（Integration Testing）、确认测试（Validation Testing）、系统测试（System Testing）和验收测试（Verification Testing）。

单元测试是针对每个单元的测试，是软件测试的最小单位，它确保每个模块能正常工作。单元测试多数使用白盒测试，用以发现内部错误。

集成测试是对已测试过的模块进行组装，进行集成测试的目的主要在于检验与软件设计相关的程序结构问题。集成测试一般通过黑盒测试方法来完成。

确认测试是检验所开发的软件能否满足所有功能和性能需求的最后手段，通常采用黑盒测试方法。

系统测试的主要任务是检测被测软件与系统的其他部分的协调性。

验收测试是软件产品质量的最后一关。这一环节，测试主要从用户的角度着手，其参与者主要是用户和少量的程序开发人员。

1.3 软件测试的生命周期

图 1-2 给出了软件测试生命周期的模型，把测试的生命周期分为几个阶段，前 3 个阶段是引入程序错误阶段，也就是开发过程中的需求规格说明、设计、编码阶段，此时极易引入错误或者导致开发过程中其他阶段产生错误。然后是通过测试发现错误的阶段，这需要通过使用一些适当的测试技术和方法来共同完成。后 3 个阶段是清除程序错误的阶段。其主要任务是进行缺陷分类、缺陷隔离和解决缺陷。其中在修复旧缺陷的时候很可能引进新的错误，导致原来能够正确执行的程序出现新的缺陷。

图 1-2 软件测试生命周期

在软件测试生命周期的每个阶段都要完成一些确定的任务，在执行每个阶段的任务时，可以采用行之有效的结构分析设计技术和适当的辅助工具；在结束每个阶段的任务时都进行严格的技术审查和管理复审。最后提交最终软件配置的一个或几个成分（文档或程序）。

1.4 软件测试与软件开发的关系

1. 测试与软件开发各阶段的关系

软件开发过程是一个自顶向下、逐步细化的过程，首先在软件计划阶段定义了软件的作用域，然后进行软件需求分析，建立软件的数据域、功能和性能需求、约束和一些有效性准则。接着进入软件开发，首先是软件设计，然后再用某种程序设计语言把设计转换成程序代码。而测试过程则是依相反的顺序安排的自底向上、逐步集成的过程，低一级测试为上一级测试准备条件。此外还有两者平行地进行测试。

如图 1-3 所示，首先对每一个程序模块进行单元测试，消除程序模块内部在逻辑上和功能上的错误和缺陷。再对照软件设计进行集成测试，检测和排除子系统（或系统）结构上的错误。随后再对照需求，进行确认测试。最后从系统全体出发，运行系统，看是否满足要求。