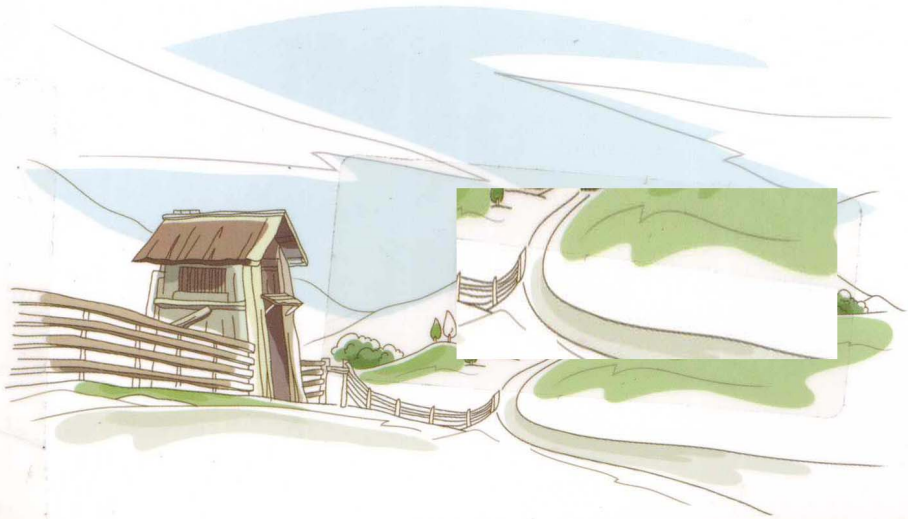
工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

# 软件测试技术教程

## Software Testing

徐光侠 韦庆杰 主编

- 采用“场景教学法”
- 理论和实践相结合
- 重视深度广度和多元性



高校系列

人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化普通高等教育“十二五”规划教材立项项目
21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

# 软件测试技术教程

## Software Testing

徐光侠 韦庆杰 主编

高校系列

人民邮电出版社
北京

**图书在版编目（CIP）数据**

软件测试技术教程 / 徐光侠，韦庆杰主编. -- 北京
人民邮电出版社，2011.4
21世纪高等学校计算机规划教材
ISBN 978-7-115-24970-8

Ⅰ. ①软… Ⅱ. ①徐… ②韦… 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(2011)第033099号

## 内 容 提 要

本书内容分为基础与实践两部分。基础部分介绍了基本概念、原理、白盒测试技术、黑盒测试技术、面向对象软件测试、单元测试、集成测试和系统测试。实践部分介绍了软件测试与软件开发过程、软件测试过程所需技能、软件测试自动化、软件测试工具、自动测试工具 QTP 等内容。

本书内容全面、深入浅出、实用性强，还易于灵活选用，适合作为高等院校计算机科学与技术专业、软件工程专业的软件测试课程的教材，也可以作为软件测试培训的教材和选择软件测试为职业的专业技术人员的参考书。

21 世纪高等学校计算机规划教材

**软件测试技术教程**

◆ 主　　编　徐光侠　韦庆杰
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：18　　2011 年 4 月第 1 版
字数：474 千字　　2011 年 4 月北京第 1 次印刷

ISBN 978-7-115-24970-8

定价：32.00 元

**读者服务热线：(010)67170985　印装质量热线：(010)67129223**
**反盗版热线：(010)67171154**

# 前 言

在大多数软件企业中，测试人员的数量不到开发人员数量的五分之一，平均比例在1:8左右，远远落后于国外先进水平。据有关调查数据显示，国内软件测试人才缺口高达30万人之多。因此，现阶段制约我国软件测试发展的一个最大的因素是人才紧缺。目前具有专业的软件测试技术工程人才极度匮乏，现有测试人员多为其他专业转行，或是培训机构培训的初级测试人员，缺乏系统的理论的基础和扎实的测试工程实践技能。

随着软件测试专业及软件测试技术课程的开设，教学和技术培训已在高校和社会软件技术培训机构中广泛开展，使软件测试课程的教学和专业人才的培养逐步有了改善。由此，对软件测试课程教材的需求也与日俱增，特别是理论阐述和工程实践结合紧密的优质教材。

本教材的编写原则和特点是：（1）理论和实践相结合。从一个软件测试工程师的角度出发，在理论基础上拓展实践技能，是贴近企业实战项目的软件测试教材。（2）注重深度、广度和多元性。全书贯穿软件工程，软件质量管理和质量保证的基础知识，让学生深刻体验软件工程开发生命周期，软件测试在软件工程中的位置，软件测试与软件质量管理、软件质量保证的关系与区别，使其具备全面的测试知识和综合的实践技能。（3）紧密结合实际。本书在讨论理论知识的同时，注重介绍这些方法在实际测试工作中的应用和实施情况，使读者在实际的测试工作中，不至于理想化地去使用各种测试方法和策略，而是时刻牢记时间、成本、质量的平衡。

本书从基础和实践两个层面引导读者学习软件测试这门学科，系统、全面地讨论了软件测试的思想、流程和方法。

第1章、第2章主要介绍软件测试的基本概念和软件测试原理。描述软件测试中基本知识以及理论基础。第3章～第7章详细介绍软件测试的核心技术，强调白盒测试、黑盒测试、面向对象测试、单元测试以及集成测试和系统测试的重点知识和相关技能以及测试用例的设计方法，让读者全面理解软件测试的实际测试方法。

第8章、第9章具体介绍软件测试与软件开发过程以及所需的技能。在技能培养方面，测试人员需加强软件测试文档的编写、软件测试用例的设计、缺陷的报告和分析，以及问题跟踪系统等各方面技能。第10章～第12章重点介绍了软件测试自动化的引入和发展，以及一些常用的商业测试工具，包括Quality Center（QC）、Quick Test Profession（QTP）、Load Runner（LR）的使用方法。第13章、第14章介绍软件测试项目阶段与测试小组的组成和自动测试实战项目案例。第13章描述了软件测试团队应该具备的架构和特点，以及如何有效地对测试团队中的测试人员进行激励和沟通，培养读者团队合作精神。第14章通过一个自动测试实战案例引导读者将学习的理论应用到实际中。

该教材的编写特色在于：（1）根据研究收集大量的国内外软件测试人才培养的教材和资料的基础上，结合当前软件工程人才培养的高质量、国际化、工程化的理念，采用大量的案例和测试工程的经验总结以简练直观的图文并茂形式，传递软件测试技术的丰富内容。其内容涉及面向过程的程序设计、面向对象的程序设计、数据结构、数据库设计与应用、操作系统等广泛学科的知识。（2）该教材采用“场景教学法”，以真实工作场景为核心，利用项目导向的角色模拟方式，规范并系统地培养专业技术人才，从实际工作内容出发，深入讲解各个基础知识点，并配备相应的图文和案例。（3）该书作为一般实践性很强的教材是编者长期在海外大型跨国企业从事软件研发，测试和服务外包的经验积累，在企业内部和软件外包公司进行专业培训的讲义的基础上，研究国内外的软件测试教材，并对互联网资料进行收集、整理和改编的结果。

本书由徐光侠、韦庆杰主编。参加编写的人员及安排如下：徐光侠编写第 1 章、第 2 章、第 3 章、第 4 章和第 12 章，金霜编写第 5 章，唐浩坤编写第 6 章，张喜平编写第 7 章，韦庆杰编写第 8 章、第 9 章、第 10 章、第 13 章和第 14 章，董滔编写第 11 章。本书在编写过程中得到了多方面的帮助、指导和支持。感谢重庆邮电大学软件学院的各位领导，他们为软件测试方向的课程规划与建设付出了大量的心血，特别感谢测试教学部的各位老师，对该书的测试案例的收集和整理付出了辛勤的劳动。

由于编者水平有限，加上软件测试领域的发展日新月异，书中难免会有疏漏和不妥之处，敬请广大读者不吝赐教。

编　者

2011 年 1 月于重庆邮电大学

# 目 录

# 第 1 章 软件测试基本概念

软件测试是伴随着软件的产生而产生的，有了软件生产和运行就必然有软件测试。软件测试就是按照测试方案和流程对产品进行功能和性能测试，甚至根据需要编写不同的测试工具，设计和维护测试系统，对测试方案可能出现的问题进行分析和评估。执行测试用例后，需要跟踪故障，以确保开发的产品适合需求。软件测试是信息系统开发中不可缺少的一个重要步骤，随着软件变得日益复杂，软件测试也变得越来越重要。

**本章内容提要**

- 什么是软件测试
- 软件测试与 CMMI
- 测试用例
- 测试环境
- 软件测试人员的需求

## 1.1 什么是软件测试

在工业制造和生产中，测试（test）被当做一个常规的检验产品质量的生产活动。测试的含义为“以检验产品是否满足需求为目标”。而软件测试活动包括了很重要的任务，即发现错误。

“软件测试”的经典定义是在规定条件下对程序进行操作，以发现错误，对软件质量进行评估。

我们知道，软件是由文档、数据以及程序组成的，那么软件测试就应该是对软件形成过程的文档、数据以及程序进行的测试，而不仅仅是对程序进行的测试。

随着人们对软件工程化的重视以及软件规模的日益扩大，软件分析、设计的作用越来越突出，而且有资料表明，60%以上的软件错误并不是程序错误，而是分析和设计错误。注意，软件错误是软件生存期内的人为错误，导致软件缺陷产生，是人为过程，相对于软件本身是外部行为。软件缺陷是存在于软件（文档、数据、程序）中的偏差，导致软件在某个特定条件下出现故障（这时称为软件缺陷被激活）。因此，做好软件需求和设计阶段的测试工作就显得非常重要。这就是我们提倡的测试概念扩大化，提倡软件全生命周期测试的理念。

# 1.2 软件测试与 CMMI

在软件开发的瀑布模型中，测试是一个非常重要的工程阶段。从保证软件质量的角度来说，软件测试是软件质量保证（软件质量保证是建立一套有计划、有系统的方法，来向管理层保证拟定出的标准、步骤、实践和方法能够正确地被所有项目所采用）工程的一个重要组成部分，也是最重要的质量保证手段。为了保证所提交的软件产品能够满足客户的需求，以及在使用中的可靠性，就必须对所开发的软件产品进行系统而全面的测试。基于这一需求，软件测试作为软件开发过程中的一个重要阶段，受到了软件开发组织的普遍重视，并形成了一整套比较成熟的测试理论和技术方法。

然而，随着软件开发技术的不断发展，以及软件系统的规模和复杂性的不断增加，传统的软件测试理论和技术已经不能够很好地满足开发组织在产品质量、开发成本、研制周期等方面的需求。本节主要从软件测试的组织和管理角度，阐述了 CMMI 模型规范对软件测试技术的应用和扩充，对于软件开发组织如何发展和完善软件开发中的测试工作进行了初步探索。

## 1.2.1 传统的软件测试技术和测试过程模型

传统的软件测试只是作为软件开发过程中的一个特定阶段，并且只针对软件成品进行测试。如图 1-1 所示，在瀑布式开发过程模型中，测试是在编码完成之后和软件产品交付运行之前的一个工程阶段，所有的审查和评审活动都是针对软件成型产品而开展。这样的软件测试主要关注的是对软件的验收测试，在一定程度上保证了所提交的软件产品的质量。但是，全面质量管理的理论认为，软件的高质量是开发和设计出来的，而不是测试出来的。因此，仅仅依靠对软件产品进行测试的质量保证活动显然是远远不够的。随着软件开发过程模型和开发技术的不断发展，软件测试理论和技术也应该得到相应的发展。

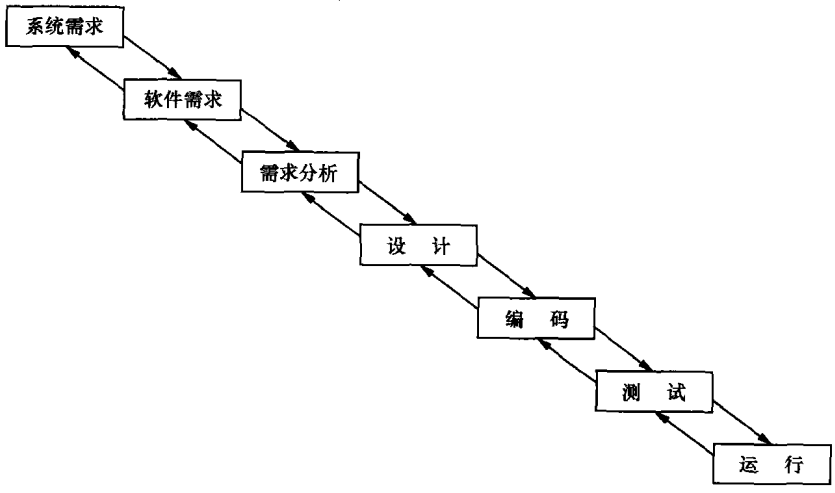



图 1-1 软件测试在软件开发过程的瀑布模型描述中所处的地位

随着全面质量管理思想在软件开发领域的应用，软件测试也由最初的只针对软件成品扩展到了针对软件半成品和过程产品的全过程测试。这是对软件测试的一种扩充。扩充后的软件测试贯穿了软件开发的全过程，包括软件需求分析、软件概要设计、软件详细设计、编码、集成、验收等各个工程阶段。相应地，各阶段所开展的测试分别为需求测试、架构测试、详细设计测试、单

元测试、集成测试以及验收测试等。这样的软件测试涵盖了软件开发的整个工程过程，对于识别和控制软件缺陷、提高软件质量起到了很明显的成效。

从本质上来说，无论是传统的软件验收测试，还是面向整个开发过程的全过程软件测试，其所针对的测试对象都是软件产品、半成品或者过程工作产品，其所报告的测试结果也只是为了识别出现在阶段产品的缺陷，并加以纠正以支持下一阶段的开发工作。从软件开发组织的长远发展来看，仅仅做到这些还是不够的。软件测试作为软件质量保证的一种重要手段，不仅要能够识别软件产品的缺陷并加以改正，还应该在软件测试中结合统计技术方法，给出对软件开发过程的度量，从而支持组织对软件开发过程的评估和改进。由美国国防部和卡耐基・梅隆大学的软件工程研究所联合开发的 CMMI 模型，正是从软件过程改进和评估的角度出发，对软件开发中的测试技术给出了充分的支持和扩充。

## 1.2.2　CMMI 模型对软件测试的支持和扩充

能力成熟度模型集成（CMMI）是由业界、美国政府和卡内基・梅隆大学软件工程研究所率先倡导的。CMMI 为改进一个组织的各种过程提供了一个单一的集成化框架，新的集成模型框架消除了各个模型的不一致性，减少了模型间的重复，增加透明度和理解，建立了一个自动的、可扩展的框架。因而能够从总体上改进组织的质量和效率。

CMMI 模型主张在开发过程中注重对过程和产品的度量，以量化的形式提供对管理过程的支持，以及对过程进行相应的评估和改进。这实际上就是对软件测试技术的一种应用和扩充。CMMI 模型将测量和分析作为一个单独的过程域，充分体现了对开发过程中的测量技术的重视，该过程域的目的就是开发和维持度量能力，以便支持对管理信息的需要。测量和分析过程域共有 3 个目标，其中两个为特定目标，一个为共性目标。

第一个目标是协调测量和分析活动。为实现这一目标，模型中给出了 4 个方面的特定实践，它们分别是确定测量对象，建立测量目标；详细说明度量值，以处理测量目标；规定数据收集和存储规程，说明如何获得并存储测量数据；规定分析规程，说明如何对度量数据进行分析和报告，并且安排优先顺序。该目标中所针对的测量对象既包括组织所开发出的软件产品、半成品以及过程产品，也包括对开发过程本身的度量。因此，在测量和分析过程中不仅要用到传统软件测试中的一些技术和方法，还需要在测量过程中引入统计过程控制等理论方法，提供对过程度量和改进的支持。

第二个目标是提供度量结果，以便处理信息需要和目标。为实现这一目标，模型中也给出了以下几方面的特定实践：收集度量数据，即获得制定的度量数据；分析并解释度量数据；管理并存储度量数据、度量规范和分析结果；通报分析结果，向所有的干系人报告测量和分析活动的结果。在这一目标中，主要关注的是对测量结果的分析和使用。在传统的软件测试中，只要产品通过了需求方的验收，达到了合同要求，开发组织一般也就不再重视对软件测试结果数据的管理和使用。从过程改进的角度来说，这是很不科学的。基于 CMMI 的集成化过程改进和评估，提出了建立开发过程数据库的思想，作为组织进行过程改进的基础。而建立过程数据的过程中，实际上也就是对测试和度量数据的积累和存储过程。从这一点来说，在开发过程中开展软件测试以及针对开发过程的度量，是建立过程数据库的必要步骤。

第三个目标是共性目标，即将测量和分析活动制度化为可管理的过程。这一目标主要关注的是对软件测试和过程度量活动的管理以及制度化。针对这一共性目标，CMMI 模型从四个不同方面给出了 10 个共性实践。首先，作为执行测量和分析活动的承诺，要求组织建立方针，为策划和执行“测量和分析”过程提供组织级的支持。其次，在执行能力方面，组织应该制定测量和分析过程计划，提供必要的资源，分配相应的责任，并且对相关人员进行培训。第三，为了指导该过

程的实施，组织应该将测量和分析过程指定的工作产品置于配置管理的适当层次，确定与过程相关的干系人并使之介入，同时还要对测量和分析过程进行监督和控制。最后，作为对测量和分析活动的验证实施，组织应该客观评价测量和分析过程以及过程的工作产品和服务的遵循情况；同时，由高层管理者审查测量和分析过程的活动、状态和分析结果，并解决相应的问题。这一共性目标的实现，实际上就是把测量和分析活动制度化为一种组织级的行为，在整个组织的范围内加强了对软件测试和过程度量活动的组织和管理工作。

从以上分析可以看出，CMMI 模型主要从以下三个方面扩充传统的软件测试技术。

（1）从单纯的对软件产品的测试活动，扩展为软件产品的测试和开发过程的度量。这一方面主要体现在过程度量对软件测试的依赖和应用。对开发过程进行度量，需要利用对软件产品、半成品以及工作产品的测试结果，从而建立对软件产品缺陷对开发过程的可跟踪性。从这一点来说，对开发过程的度量，实际上也就是针对软件产品的测试活动的扩展，其与传统的软件测试的不同之处就在于关注对软件测试结果数据的分析和利用，将测试数据有效转换成为能够标识过程缺陷的统计数据。

（2）软件测试由原来的事后测试行为发展为全过程测试和分析，成为一种缺陷预防的有效方式。统计技术方法的应用，将传统的软件测试活动扩展为一种全过程测试行为。从质量工程的角度来说，这是一种质量保证思想的转变。传统的软件测试，只针对软件产品而开展，找到缺陷之后再加以改正和修补，这是一种“亡羊补牢”的质量管理方式；而针对开发全过程所开展的软件测试和过程度量，则注重根据对测试数据的统计分析结果，来判断软件产品的未来质量趋势，并提前予以控制和预防，属于一种“防患于未然”的质量管理方式。与传统的软件测试相比，全过程测试不仅可以有效降低产品的质量风险，而且还可以提前对软件产品缺陷进行规避，这不仅缩短了对缺陷的反馈周期和整个项目的开发周期，而且也大大降低了对软件产品的维修和维护费用，对于软件产品的整个生命周期都有很重大的意义。

（3）软件测试与开发过程的其他阶段不再是串行工作方式，而是与整个开发过程并行进行。与瀑布模型相比，CMMI 模型中所描述的软件测试和过程度量工作与整个开发过程是并行进行的，是一种基于并行工程的测试和度量行为。基于并行工程开展的软件测试活动，存在于软件生命周期的各个阶段，其基本特点是以质量保证和客户要求为核心开展对软件产品和开发过程的测试和度量，力争将缺陷控制在软件开发过程的每一个阶段，从而可以有效缩短开发周期，降低质量风险，并且可以及时吸取经验教训，提供对过程改进的支持。这也体现了 CMMI 模型对并行工程思想的一种支持和应用。

除了测量和分析过程域之外，CMMI4 中的量化过程管理过程域也是对软件测试和过程度量技术的一种更高层次上的应用。在该过程域中，测试和度量已经不仅仅是一种被管理的过程，而且其本身也成为了一种有效的辅助管理手段，从定量化的角度对软件开发过程的管理和组织活动给出了支持。开发过程管理从定性到定量的转化，是 CMMI 集成化过程改进所追求的目标之一，也是开发组织一直追求的一种更高水平的管理方式。因此，随着软件开发组织过程能力的不断提高，软件测试和度量技术也将会得到不断的发展和完善。

# 1.3 测试用例

## 1.3.1 什么是测试用例

测试用例（Test Case，TC）简单来讲是指执行条件和预期结果的集合，完整来讲是针对要测

试的内容所确定的一组输入信息，是为达到最佳的测试效果或高效地揭露隐藏的错误而精心设计的少量测试数据。

RUP（Rational Unified Process，统一软件开发过程）中认为测试用例是我们用来验证系统实际做了什么的方式，因此，测试用例必须可以按照要求来跟踪和维护。

IEEE 标准 610（1990）给出的定义为：测试用例是一组测试输入、执行条件和预期结果的集合，目的是要满足一个特定的目标，比如执行一条特定的程序路径或检验是否符合一个特定的需求。

从以上定义来看，测试用例设计的核心有两方面，一是要测试的内容，即与测试用例相对应的测试需求；二是输入信息，即按照怎样的操作步骤，对系统输入哪些必要的数据。测试用例设计的难点在于如何通过少量测试数据来有效揭示软件缺陷。

测试用例可以用一个简单的公式来表示：

测试用例= 输入 + 输出 + 测试环境

其中，输入是指测试数据和操作步骤，输出是指系统的预期结果，测试环境是指系统环境设置，包括软件环境、硬件环境和数据，有时还包括网络环境。

## 1.3.2　测试用例的评价标准

根据多年的实践经验，测试用例的标准不能局限于一个层次，因为测试用例设计类似于软件设计，软件设计有架构设计（结构设计/概要设计）和详细设计，所以对于测试用例的质量标准，也应分为如下两个层次来考虑。

（1）高层次——以满足某一个测试目标或测试任务来整体看测试用例，衡量一组测试用例的结构、设计思路和覆盖率等指标。

（2）低层次——从单个测试用例看，衡量其描述的规范性、可理解性和可维护性等指标。

1. 高层次（high-level）标准

高层次标准是从满足某一个特定的测试目标出发来进行定义，分析一组测试用例的设计思路、设计方法和策略，包括测试用例的层次、结构等。从高层次看，测试用例设计的关键点是：始终从客户需求的角度出发，始终围绕测试的覆盖率和执行效率不断思考，最终通过有效的技术方法完成测试用例的设计。

对于一整套的测试用例组（集合），可定义如下的质量标准。

（1）测试用例的目标清楚，并能满足软件质量的各个方面，包括功能测试、性能测试、安全性测试、故障转移测试、负载测试等。

（2）设计思路正确、清晰。例如，通过序列图、状态图、工作流程图、数据流程图等来描述待测试的功能特性或非功能特性。

（3）在组织和分类上，测试用例层次清楚、结构合理。测试用例的层次与产品特性的结构/层次相一致，或者与测试的目标/子目标的分类/层次相一致，并具有合理的优先级或执行顺序。

（4）测试用例覆盖所有测试点、覆盖所有已知的用户使用场景（User scenario），也就是说每个测试点都有相应数量的测试用例来覆盖，而且将各种用户使用场景通过矩阵或因果图等方式列出来，找到相对应的测试用例。

（5）测试手段的区别对待。在设计测试用例时，就要全面考量测试的手段，哪些方面可以通过工具测试，哪些方面不得不用手工测试，对不同手段的测试用例区别对待。

（6）有充分的负面测试。作为测试用例，不仅要测试正确的输入和操作，还要测试各种各样的例外情况，如边界条件、不正确的操作、错误的数据输入等。

（7）没有重复、冗余的测试用例，满足相应的行业标准等。

2. 低层次（low-level）标准

低层次标准是考察单个测试用例是否满足测试的需求，是否能被更有效地执行。测试用例设计的结果就是交付测试用例，使测试用例被执行，所以除了覆盖率，执行的效率也是测试用例的一个重要属性。测试用例越清楚，越容易被理解和执行。执行效率越高就说明测试用例越好，如果测试用例能被机器（computer）执行，当然执行效率得到体现。

对于具体的某个测试用例，不妨可定义如下的质量标准。

（1）测试用例的出发点是发现缺陷，即单个测试用例在“暴露缺陷”上具有较高的可能性。

（2）测试用例的单一性。一个测试用例面向一个测试点，不要将许多测试点揉在一起。例如，通过一个测试用例发现1～2个缺陷，而不能发现5～10个缺陷甚至更多的缺陷。

（3）符合测试用例设计规范或测试用例模板。

（4）描述清楚。包括特定的场合、特定的对象和特定的术语，没有含糊的概念和一般性的描述。例如，测试用例名称为“登录功能使用正常”，就是一个描述不清楚的例子，而这样的描述“登录功能中用户名大小写不敏感性验证”、“登录功能中用户名唯一性验证”和“用户账号被锁定后再进行登录操作”等就比较好。

（5）操作步骤的准确性。按照步骤的操作得到唯一的测试结果。

（6）操作步骤的简单性。操作步骤不应该太复杂，过于复杂的操作步骤意味着测试用例需要被分解为多个测试用例或者分解为多个环节进行验证。

（7）所期望的测试结果是可验证的，即能迅速、明确地判断测试的实际结果是否与所期望的结果相同或相匹配。例如，在测试用例中描述期望结果为“登录成功”，这实际是不可验证的。要使这个期望结果具有可验证性，我们就应该这样描述所期望的结果——“‘退出（log out）’按钮出现”。

（8）测试环境的正确性、测试数据的充分性。

（9）前提条件、依赖性被完全识别出来。

这样，测试用例具有很好的可理解性和可维护性，可以提高测试执行的效率。并能保证不同的人员执行相同的用例能获得统一的结果。步骤的准确性和期望结果的可验证性，有助于测试执行的自动化实现。也只有实现了测试执行的自动化，测试执行的效率才是最高的，而且测试人员才有更多的时间去思考、去设计更优秀的测试用例，进入良性循环，相互促进，不断地提升测试的质量和效率。

### 1.3.3 测试用例设计的基本原则

对于不同类别的软件，测试用例的设计重点是不同的。比如，公司管理软件的测试通常需要将测试数据和测试脚本从测试用例中划分出来，测试用例更趋于是一种针对软件产品的功能、业务规则和业务处理所设计的测试方案，对软件的每个特定功能或运行操作路径的测试构成不同的测试用例。

一般情况下，测试用例设计的基本原则有3条。

（1）测试用例的代表性。能够代表并覆盖各种合理的和不合理的、合法的和非法的、边界的和越界的以及极限的输入数据、操作和环境设置等。

（2）测试结果的可判定性。即测试执行结果的正确性是可判定的，每一个测试用例都应有相应明确的预期结果，而不应存在二义性，否则将难以判断系统是否运行正常。

（3）测试结果的可再现性。即对同样的测试用例，系统的执行结果应当相同。测试结果可再

现有利于在出现缺陷的时候能够确保缺陷的重现，为缺陷的快速修复打下基础。

而以上 3 条原则中，最难保证的就是测试用例的代表性，这也是我们设计测试用例时最为关注的内容，即如何确定测试用例中关于输入的数据集合。一般地，在有多个输入条件的情况下，应首先分析出哪些是核心的输入条件。针对每个核心的输入条件，其数据大致可分为以下 3 类。

① 正常数据。符合需求规格说明，合理的、有效的输入数据，如某个输入的有效取值范围内的数据。

② 边界数据。介于正常数据与错误数据之间的临界数据。边界数据可能是有效的输入数据，也有可能是无效的输入数据，这要根据需求规格说明的具体规定而定。

③ 错误数据。不符合需求规格说明，无意义的、无效的输入数据。注意，对于无效输入有两种情况：第一，满足所有输入的数据类型要求，但不在有效取值范围内；第二，部分输入不满足输入的数据类型要求。更严格地，还应考虑缺少输入条件的情况。

测试数据就是从以上 3 类数据中产生的。

### 1.3.4　测试用例模板

通过以上学习，相信大家都大致对测试用例有了一定的了解，那么下面将向大家展示具体的测试用例的模板，大家可以通过学习用例模板，在今后的学习工作当中灵活运用。

一个优秀的测试用例，应该包含以下信息。

（1）软件或项目的名称。

（2）软件或项目的版本（内部版本号）。

（3）功能模块名。

（4）测试用例的简单描述，即该用例执行的目的或方法。

（5）测试用例的参考信息（便于跟踪和参考）。

（6）本测试用例与其他测试用例间的依赖关系。

（7）本用例的前置条件，即执行本用例必须要满足的条件，如对数据库的访问权限。

（8）用例的编号（ID），如可以是“软件名称简写—功能块简写—NO.”。

（9）步骤号、操作步骤描述、测试数据描述。

（10）预期结果（这是最重要的）和实际结果（如果有缺陷管理工具，这条可以省略）。

（11）开发人员（必须有）和测试人员（可有可无）。

（12）测试执行日期。

表 1-1 所示模板就是一个测试用例模板。

表 1-1　　测试用例模板

| 项目/软件 | 技术出口合同网络申领系统 | 程序版本 | 1.0.25 | | | |
|---|---|---|---|---|---|---|
| 功能模块名 | Login | 编制人 | xxx | | | |
| 用例编号 | TC-TEP_Login_1 | 编制时间 | 2010.10.12 | | | |
| 相关的用例 | 无 | | | | | |
| 功能特性 | 用户身份验证 | | | | | |
| 测试目的 | 验证是否输入合法的信息，允许合法登录，阻止非法登录 | | | | | |

续表

| 预置条件 | 无 | 特殊规程说明 | 如数据库访问权限 | | | |
|---|---|---|---|---|---|---|
| 参考信息 | 需求说明中关于“登录”的说明 | | | | | |
| 测试数据 | 用户名 =yiyh 密码 =1 | | | | | |
| 操作步骤 | 操作描述 | 数据 | 期望结果 | 实际结果 | 测试状态 | |
| 1 | 输入用户名称，按“登录”按钮 | 用户名=yiyh，密码为空 | 显示警告信息“请输入用户名和密码！” | | | |
| 2 | 输入密码，按“登录”按钮 | 用户名为空，密码=1 | 显示警告信息“请输入用户名和密码！” | | | |
| ..... | | | | | | |
| 测试人员 | | 开发人员 | | | 项目负责人 | |

# 1.4 测 试 环 境

## 1.4.1 什么是测试环境

软件测试环境就是软件运行的平台，即软件、硬件和网络的集合，如下式所示：

测试环境 = 软件 + 硬件 + 网络+ 历史数据

- 硬件：主要包括 PC、笔记本电脑、服务器、各种 PDA 终端等。不同的机器类型，不同的配置，会导致程序的反应速度完全不一样，所以我们测试一款软件时一定要考虑硬件配置的因素。
- 软件：这里主要指的是软件运行的操作系统。许多软件在使用时有兼容性的问题。
- 网络：主要针对的是 C/S 结构和 B/S 结构的软件。
- 历史数据：指测试用例执行所需初始化的各项数据。

以上就是我们搭建软件测试环境需要考虑的 3 个方面，作为一名合格的软件测试工程师，不仅要熟悉软件的知识，也要了解硬件和网络的相关知识。

## 1.4.2 测试环境的规划

一般情况下，单元测试和集成测试由开发人员在开发环境中进行，验收测试主要在用户最终应用环境中进行，在此暂不讨论。因此，我们需要搭建的环境主要用于被测软件系统（包括 Web 应用）的系统测试。

规划测试环境的第一步是明确如下问题。

（1）执行测试所需的计算机数量和对每台计算机的硬件配置要求，包括 CPU 速度、硬盘和内存容量、网卡支持的速度等。

（2）部署服务器所需的操作系统、数据库管理系统（DBMS）、中间件、Web 服务器等（以下统称支撑软件环境）的名称、版本，必要时还需明确相关补丁的版本。

（3）用于保存文档和数据（这里主要是指测试过程中生成的文档，而非测试参考文档或存放测试结果的最终文档）的服务器必需的支撑软件环境中各软件的名称、版本，必要时也应明确相关补丁的版本。