高职高专教育计算机系列教材

计算机软件技术类

- Visual FoxPro程序设计
- Visual Basic程序设计
- Visual Basic.NET程序设计
- ASP.NET Web程序设计
- C++程序设计
- Visual C++程序设计教程
- C#程序设计
- XML基础教程
- Java程序设计
- JavaScript程序设计
- JSP程序设计
- Java ME无线开发实用教程
- Delphi程序设计
- SQL Server 2005实用教程
- Oracle数据库应用教程
- 软件测试

策划编辑：刘　建
责任编辑：唐卫威
装帧设计：蒋宏工作室

ISBN 978-7-81099-820-8
9 787810 998208 03 >

定价：42.00元

高职高专教育计算机系列教材

软件测试

国防科技大学出版社

高职高专教育计算机系列教材

软件测试

主编　徐光侠

国防科技大学出版社

高职高专教育计算机系列教材

软件测试

主　编　徐光侠
副主编　黄海辉　王佳榕
　　周　林　鲁守玮

国防科技大学出版社

【内容简介】本教材是为高职高专计算机及相关专业编写的教材。

本书较系统地介绍了软件测试技术中的基本概念、基本原理以及常用方法，并将近些年出现的一些新技术加以融合。本书共分为11章，内容包括软件测试概述、软件测试原理、黑盒测试、白盒测试、面向对象软件的测试、单元测试、集成测试、系统测试、验收测试和回归测试、软件测试计划和测试文档以及软件测试实例。本书结合相关实例说明，较为全面地阐述了软件测试相关技术。

本教材适合高职高专学生使用，也可供相关技术人员参考。

图书在版编目(CIP)数据

软件测试/徐光侠主编. —长沙：国防科技大学出版社，2010.12(2026.3重印)

ISBN 978-7-81099-820-8

Ⅰ. ①软… Ⅱ. ①徐… Ⅲ. ①软件—测试 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2010)第221651号

出版发行：国防科技大学出版社
责任编辑：唐卫威
印 刷 者：三河市骏杰印刷有限公司
开　　本：787mm×1 092mm　1/16
印　　张：13.75
字　　数：341千字
版　　次：2026年3月第1版第11次印刷
定　　价：42.00元

编审委员会

出 版 说 明

高职高专教育作为我国高等教育的重要组成部分，承担着培养高素质技术、技能型人才的重任。近年来，在国家和社会的支持下，我国的高职高专教育取得了不小的成就，但随着我国经济的腾飞，高技能人才的缺乏越来越成为影响我国经济进一步快速健康发展的瓶颈。这一现状对于我国高职高专教育的改革和发展而言，既是挑战，更是机遇。

要加快高职高专教育改革和发展的步伐，就必须对课程体系和教学模式等问题进行探索。在这个过程中，教材的建设与改革无疑起着至关重要的基础性作用，高质量的教材是培养高素质人才的保证。高职高专教材作为体现高职高专教育特色的知识载体和教学的基本工具，直接关系到高职高专教育能否为社会培养并输送符合要求的高技能人才。

为促进高职高专教育的发展，加强教材建设，教育部在《关于全面提高高等职业教育教学质量的若干意见》中，提出了“重点建设好 3 000 种左右国家规划教材”的建议和要求，并对高职高专教材的修订提出了一定的标准。为了顺应当前我国高职高专教育的发展潮流，推动高职高专教材的建设，我们精心组织了一批具有丰富教学和科研经验的人员成立了编审委员会。

编审委员会依据教育部制定的《高职高专教育基础课程教学基本要求》和《高职高专教育专业人才培养目标及规格》，调研了百余所具有代表性的高等职业技术学院和高等专科学校，广泛而深入地了解了高职高专的专业和课程设置，系统地研究了课程的体系结构，同时充分汲取各院校在探索培养应用型人才方面取得的成功经验，并在教材出版的各个环节设置专业的审定人员进行严格审查，从而确保了整套教材“突出行业需求，突出职业的核心能力”的特色。

本套教材的编写遵循以下原则：

(1) 成立教材编审委员会，由编审委员会进行教材的规划与评审。

(2) 按照人才培养方案以及教学大纲的需要，严格遵循高职高专院校各学科的专业规范，同时最大程度地体现高职高专教育的特点及时代发展的要求。因此，本套教材非常注重培养学生的实践技能，力避传统教材“全而深”的教学模式，将“教、学、做”有机地融为一体，在教给学生知识的同时，强化了对学生实际操作能力的培养。

(3) 教材的定位更加强调“以就业为导向”，因此也更为科学。教育部对我国的高职高专教育提出了“以应用为目的，以必需、够用为度”的原则。根据这一原则，本套教材在编写过程中，力求从实际应用的需要出发，尽量减少枯燥、实用性不强的理论灌输，充分体现出“以行业为向导，以能力为本，以学生为中心”的风格，从而使本套教材更具实用性和前瞻性，与就业市场结合也更为紧密。

(4) 采用“以案例导入教学”的编写模式。本套教材力图突破陈旧的教育理念，在讲解的过程中，援引大量鲜明实用的案例进行分析，紧密结合实际，以达到编写实训教材的

目标。这些精心设计的案例不但可以方便教师授课，同时又可以启发学生思考，加快对学生实践能力的培养，改革人才的培养模式。

本套教材涵盖了公共基础课系列、财经管理系列、物流管理系列、电子商务系列、计算机系列、电子信息系列、机械系列、汽车系列和化学化工系列的主要课程。

对于教材出版及使用过程中遇到的各种问题，欢迎您及时与我们取得联系。同时，我们希望有更多经验丰富的教师加入到我们的行列当中，编写出更多符合高职高专教学需要的高质量教材，为我国的高职高专教育做出积极的贡献。

编审委员会

序

21世纪是科技和经济高速发展的重要时期。随着我国经济的持续快速健康发展，各行各业对高技能专业型人才的需求量迅速增加，对人才素质的要求也越来越高。高职高专教育作为我国高等教育的重要组成部分，在加快培养高技能专业型人才方面发挥着重要的作用。

与国外相比，我国高职高专教育起步晚，发展时间短，这种状况与我国经济发展对人才大量需求的现状是很不协调的。因此，必须加快高职高专教育的发展步伐，提高应用型人才的培养水平。

高职高专教育水平的提高，离不开课程体系的完善。相关领域人才的培养需要一批兼具前瞻性和实践性的优秀教材。教育部职业教育与成人教育司针对高职高专教育人才培养模式提出了“以就业为导向”的指导思想，这也正是本套高职高专教材的编写宗旨和依据。

如何使高职高专教材既突出行业的需求特点，又突出职业的核心能力？这是教材编写的过程中必须首先解决的问题。本系列教材编委会深入研究了高职高专教育的课程和专业设置，并对以往的教材进行了详细分析和认真考察，力图在不破坏教材系统性的前提下，加强教材的创新和实践性内容，从而确保学生在学习专业知识的同时多动手，增强自己的实践能力，以加强“知”与“行”的结合。

同时，本系列教材在编写过程中还充分重视群体和类别的差异性，面对不同学校和不同专业方向的定位差异，精心设计了一些有代表性的例题和课后习题，以满足不同的需求。另外，本系列教材设计了配套的精品教学资料包，包含教学课件、教学参考资料、课后习题答案等丰富的教学资源。

经过编委会的辛勤努力，本套教材终于顺利出版了，相信本套教材一定能够很好地适应现代高职高专教育的教学需求，也一定能够在高职高专教育计算机课程的改革中发挥积极的推动作用，为社会培养更多优秀的应用型人才。

全国计算机基础教育研究会副秘书长 

前　言

随着计算机技术的发展以及计算机应用领域的不断扩大，计算机系统的规模和复杂性急剧增加，软件开发成本及由于软件故障而造成的经济损失日益增加。由于这些问题越来越突出，使得软件质量问题逐渐成为人们关注的焦点。

软件测试是一门新兴学科，目前研究的内容并不十分深入，仍然处于起步阶段。软件测试是保证软件质量的重要手段。通过软件测试，测试人员可以尽可能地发现软件中存在的漏洞，从而修复弥补，为软件质量的提高提供帮助。

软件测试是一项专业性较强的工作，它包括软件需求分析、设计规格说明和编码的最终评审等许多理论知识，同时要求软件测试人员具有一定的工程实践经验。因此，软件测试人员需要接受专门的培训和学习，并在实践中不断积累经验，从而成为一名合格的软件测试人员。

本书内容涵盖了软件测试的各项技术和基础知识。在编写过程中，我们注意保持教材内容的先进性，将软件测试的新概念、新理论、新技术以及新方法融入其中，内容的编排由易到难，注重理论与实践相结合。

全书共分为 11 章，第 1 章是软件测试概述，包括软件开发与软件测试的关系、缺陷管理以及软件测试职业的现状等；第 2 章介绍了软件测试原理，从软件测试的原则、软件测试的分类、软件测试的过程以及软件测试的过程模型等几个方面论述了软件测试的原理；第 3 章主要介绍了黑盒测试的优点和缺点及其所使用的方法和工具；第 4 章讲解了白盒测试；第 5 章介绍了面向对象软件的测试技术；第 6 章介绍了单元测试；第 7 章讲解了集成测试的相关知识和案例；第 8 章介绍了系统测试，内容包括系统测试概述、系统测试的类型及过程等；第 9 章介绍了验收测试和回归测试的相关知识，并介绍了回归测试的工具；第 10 章详细介绍了软件测试计划的制订以及软件测试文档的编写；第 11 章是一个较为全面的软件测试实例，整体演示了软件测试的过程。

本书由徐光侠任主编，黄海辉、王佳榕和周林任副主编。参加编写的人员及分工安排如下：徐光侠编写第 1 章，李宁编写第 2 章和第 3 章，黄海辉编写第 4 章，鲁守玮编写第 5 章、第 6 章和第 9 章，刘姝红编写第 7 章和第 8 章，周林编写第 10 章，万宏凤编写第 11 章。感谢何远、黄明辉、年欢、刘睿、宋彦、李建秋为本书做的相关文字处理、插图等工作。

由于编者水平有限，加之时间仓促，书中难免出现疏漏和不足之处，敬请广大读者批评指正。

编　者

目　录

第 1 章　软件测试概述

知识目标

◎ 软件开发与软件测试
◎ 软件测试与 CMMI
◎ 缺陷管理、测试用例和测试环境
◎ 软件测试职业

技能目标

◎ 了解软件测试的定义和重要性
◎ 理解软件测试技术的概念及 CMMI 模型的作用
◎ 掌握 BUG 的定义、分类、处理流程，以及缺陷报告的撰写形式
◎ 掌握测试用例的定义、评价标准及其设计的基本原则
◎ 掌握测试环境的定义、要素、规划及维护和处理
◎ 了解国内外软件测试的现状、软件测试人员的构成及素质要求

软件测试就是利用测试工具按照测试方案和流程对产品进行功能和性能测试，或是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。

1.1　软件开发与软件测试

软件开发与软件测试是两个相互联系的概念，软件测试贯穿于软件开发的整个生命周期。

1.1.1　软件开发

1. 软件开发的定义

软件开发是一个把用户需要转化为软件需求，把软件需求转化为软件设计，用软件代码来实现软件设计，对软件代码进行测试，并确认它可以投入运行使用的过程。在这个过程中的每一阶段，都包含有相应的文档编制工作。

2. 软件开发过程

1)需求分析

软件需求分析就是回答系统“做什么”的问题。它是一个对用户的需求进行去粗取精、去伪存真,正确理解,然后把它用软件工程开发语言(形式功能规约,即需求规格说明书)表达出来的过程。本阶段的基本任务是和用户一起确定要解决的问题,建立软件的逻辑模型,编写需求规格说明书文档并最终得到用户的认可。需求分析的主要方法有结构化分析方法、数据流程图和数据字典等。本阶段的工作是根据需求规格说明书的要求,设计建立相应的软件系统的体系结构,并将整个系统分解成若干个子系统或模块,定义子系统或模块间的接口关系,对各子系统进行具体设计定义,编写软件概要设计说明书和详细设计说明书、数据库或数据结构设计说明书、组装测试计划等。

2)设计

软件设计可以分为概要设计和详细设计两个阶段。实际上软件设计的主要任务就是将软件分解成模块,即能实现某个功能的数据和程序说明、可执行程序的程序单元。这些程序单元可以是一个函数、过程、子程序,也可以是可组合、可分解和可更换的功能单元。

概要设计就是结构设计,其主要目的是给出软件的模块结构,用软件结构图表示。

详细设计的首要任务是设计模块的程序流程、算法和数据结构,次要任务是设计数据库,常用方法是结构化程序设计方法。

3)编码

软件编码是指把软件设计转换成计算机可以接受的程序,即写成以某一程序设计语言表示的“源程序清单”。充分了解软件开发语言、工具的特性和编程风格,有助于开发工具的选择以及保证软件产品的开发质量。

当前软件开发中除专用场合外,已经很少使用 20 世纪 80 年代流行的计算机高级语言了,取而代之的是面向对象的开发语言,而且面向对象的开发语言和开发环境大都合为一体,这就大大提高了开发的效率。

4)测试

软件测试的目的是以较小的代价发现尽可能多的错误。实现这个目标的关键在于设计一套出色的测试用例(测试用例由测试数据和预期的输出结果组成)。如何才能设计出一套出色的测试用例? 关键在于理解测试方法。不同的测试方法有不同的测试用例设计方法。常用的两种测试方法是白盒法和黑盒法。

白盒法的测试对象是源程序,依据程序内部的逻辑结构来发现软件的编程错误、结构错误和数据错误。结构错误包括逻辑、数据流、初始化等的错误。白盒法用例设计的关键是以较少的用例覆盖尽可能多的内部程序逻辑结果。

黑盒法依据软件的功能或软件行为描述来发现软件的接口、功能和结构错误。其中接口错误包括内部/外部接口、资源管理、集成化以及系统错误。黑盒法用例设计的关键是以较少的用例覆盖模块输出和输入接口。

5)维护

维护是指在已完成对软件的研制(需求分析、设计、编码和测试)工作并交付使用以后,对软件产品所展开的一些软件工程活动,即根据软件运行的情况,对软件进行适当修改,以适应新的要求,以及纠正运行中发现的错误,最后,还需编写软件问题报告、软件修改报告。

一个中等规模的软件,如果研制阶段需要 1～2 年的时间,在它投入使用以后,其运行或

工作时间可能持续 5～10 年，那么它的维护阶段即在运行的这 5～10 年期间。这段时间，软件维护人员需要着手解决研制阶段所遗留的各种问题，同时还要解决某些维护工作本身所特有的问题。做好软件维护工作，不仅能排除障碍，使软件能正常工作，而且还可以扩展功能，提高性能，为用户带来明显的经济效益。然而遗憾的是，现在人们对软件维护工作的重视往往远不如对软件研制工作的重视。而事实上，和软件研制工作相比，软件维护的工作量和成本都要大得多。

在实际开发过程中，软件开发并不是从第一步按顺序进行到最后一步，而是在任何阶段，在进入下一阶段前一般都有一步或几步的回溯。测试过程中发现的问题可能要求修改设计，用户可能会提出一些需求来修改需求说明书等。

1.1.2　软件测试

1. 软件测试的定义

软件测试是指使用人工和自动手段来运行或测试某个系统的过程，目的在于检验其是否满足规定的需要或弄清楚预期结果与实际结果之间的差别。

2. 软件测试的重要性

在软件业比较发达的国家，软件测试不仅早已成为软件开发的一个有机组成部分，而且在整个软件开发的系统工程中占据着相当大的比重，软件测试是十分重要的。具体而言，软件测试的重要性体现在以下几个方面。

1）寻找软件错误，以便进行修正

这是保证软件质量的重要一环，也是测试人员需要持续不断做的工作，特别是通过回归测试可以防止无意识的行为引入一些将来可能出现的错误。

2）验证软件是否符合要求

从用户的角度出发，用户希望通过测试来核实开发方交付的软件是否符合自己的真实需求，系统运行在真实的应用环境下是否存在关键功能或关键性能上的缺陷，这主要是指 β 测试。在测试过程中，用户的目标是全方位使用系统，尽可能多地暴露软件中的问题，以考虑是否接受该产品。实施良好的 β 测试非常有利于产品的成功发布。

3）证明软件符合要求，是可用的

从开发方的角度出发，开发方希望通过测试向用户证明其所开发的软件正确地实现了用户需求，树立用户对软件质量的信心，为用户选择与接受软件提供有力的依据，这主要是指 α 测试。这时，开发方的目标是确保程序不要出大的问题，避免延误产品的正常交付。

4）指导软件的开发过程

就宏观而言，不同的软件开发过程，有不同的测试模型与之对应，以此能够更好地指导开发与测试实践；就微观来看，测试可以保证对需求和设计的理解与表达的正确性、实现的正确性及运行的正确性，无论哪个环节存在问题，都将在测试中表现出来。通过分析缺陷的分布和产生原因可以帮助测试人员发现当前软件开发的缺陷，有助于过程改进，通过分析整理测试结果，可以进一步完善软件。

5）提供软件的相关特征

测试是对软件质量的度量和评估。测试可提供测试数据及结果来对质量进行细化和量化。

软件开发与软件测试是相互联系的，软件开发过程中不能缺少软件测试这一环节，软件测试应当伴随着软件开发的整个过程。

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工程的一个重要组成部分，也是最重要的保证手段。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很好地满足开发组织在产品质量、开发成本以及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 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成型软件产品而开展的。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，仅仅依靠对软件成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。

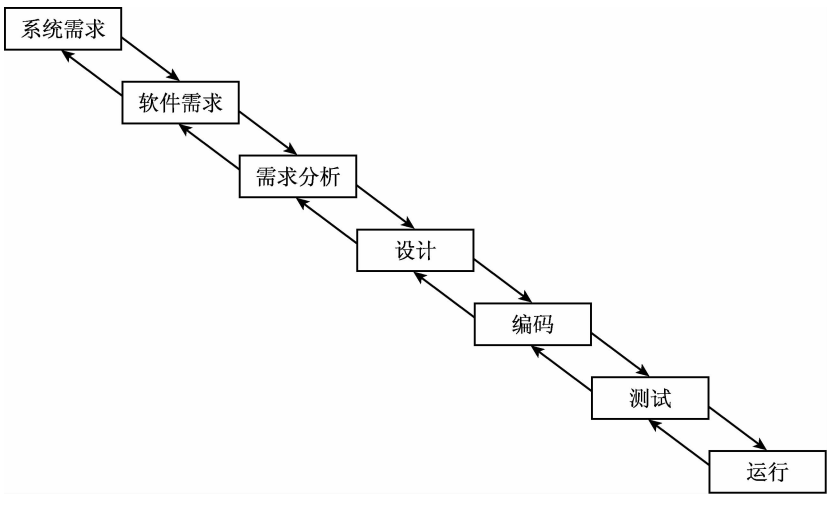

图 1-1 软件测试在软件开发过程的瀑布模型描述中所处的地位

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、提高软件质量有很明显的效果。

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成品、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

CMMI 模型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，以便对管理信息的需要提供支持。

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成品、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，提供对过程度量的改进和支持。

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 4 个方面的特定实践：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，向所有的相关人员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。而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。CMMI 对集成化过程进行了改进和评估，提出了建立开发过程数据库的思想，以作为开发组织进行过程改进的基础。而建立过程数据库的过程，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 4 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，组织应该制订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，确定与过程相关的人员并使之介入，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 3 个方面扩充了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，需要利用对软件成品、半成品以及工作产品的测试结果，从而建立对开发过程的软件产品缺陷可跟踪性。从这一点来说，对开发过程的度量，实际上也就是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它关注对软件测试结果数据的分析和利用，将测试数据有效地转换成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，只针对软件成品而开展，找到缺陷之后再加以改正和修补，这是一种“亡羊补牢”式的质量管理方式；而针对开发全过程所开展的软件测试和过程度量，则根据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，并提前予以预防和控制，属于一种“防患于未然”式的质量管理方式。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修改和维护费用，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，CMMI 4 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　缺 陷 管 理

缺陷管理(defect management)是在软件生命周期中获取、管理、沟通任何变更请求的过程(从变更的建议到变更的解决)。缺陷管理可以确保问题(如需求或者缺陷)被跟踪管理而不丢失。在程序中存在的软件缺陷(如文档缺陷、代码缺陷、测试缺陷、过程缺陷)导致系统或部件不能实现其功能，引起系统的失效。对软件缺陷测试和管理是不可缺少的。

1.3.1　BUG 的定义与分类

缺陷(BUG)是指程序中存在的错误，如语法错误、拼写错误或者一个不正确的程序语句。缺陷可能出现在设计中，或是在需求分析、规格说明或其他文档中。软件缺陷导致系统或部件不能实现其功能，引起系统的失效。缺陷定义如下：

- 软件没有实现产品说明书标明的功能。
- 程序中存在语法错误。
- 程序中存在拼写错误。
- 程序中存在不正确的程序语句。
- 软件出现了与产品说明书不一致的表现。
- 软件功能超出产品说明书的范围。
- 软件没有达到用户期望的目标。
- 测试人员或用户认为软件的易用性差。

从不同的分类角度，可以将 BUG 分为多种类型。

1)按严重程度划分

严重程度是指 BUG 对软件质量的破坏程度，即此 BUG 的存在将对软件的功能和性能产生怎样的影响。按照严重程度由高到低的顺序可以将其分成 5 个等级：系统崩溃、严重、一般、次要、建议。需要说明的是，在具体的项目中，要根据实际情况来划分等级，不一定是 5 个等级，如果 BUG 数比较少，就可以划分为 3 个等级：严重、一般、次要。一般的缺陷管理工具会自动给出一个默认的 BUG 严重程度划分。

2)按优先级划分

优先级是指处理和修复软件缺陷的先后顺序的指标，即哪些缺陷需要优先修正，哪些缺陷可以稍后修正。按照优先级由高到低的顺序可以将其分成 3 个等级：高(high)、中(middle)、低(low)。其中高优先级的 BUG 是应该立即修复的 BUG，中优先级的 BUG 是应该在产品发布之前修复的 BUG，低优先级的 BUG 是指如果时间允许应该修复的 BUG 或是可以暂时存在的 BUG。和 BUG 的严重程度的划分一样，优先级的划分方法也不是绝对的，需要根据实际情况灵活划分。

3)按照测试种类划分

按照测试种类可以将 BUG 分为逻辑功能类(function)、性能类(performance)、界面类(UI)、易用性类(usability)、兼容性类(compatibility)等。可以这样理解：有一种测试方法，

就有一种对应的 BUG 种类。当然这里列举的都是黑盒测试的测试方法，还有白盒测试的测试方法也可以作为 BUG 的种类，如边界值类、内存溢出类、逻辑驱动类等。

4)按功能模块划分

一般的软件产品都是分为若干个功能模块的，如在 Word 2000 中有文件、编辑视图、帮助等功能模块。80%的缺陷集中在 20%的模块里，测试时，可以统计一下 BUG 主要集中在哪些模块里面，以便投入重点精力去测试。

5)按 BUG 的生命周期划分

可以把 BUG 看做是一个有生命的“小虫子”，每个 BUG 都有其生命周期，可以这样划分：新建(new)、确认(confirmed)、解决(fixed)、关闭(closed)、重新打开(reopen)。

1.3.2 缺陷报告

在实际项目中，需要根据固定的模板提交 BUG，这个模板可以是 Word、Excel 或是缺陷管理工具自带的模板，其基本形式如表 1-1 所示。

表 1-1 缺陷报告模板

<table>
<tr><td colspan="2">缺陷记录</td><td>编号：0001</td></tr>
<tr><td>软件名称：××字处理软件</td><td>模块名：帮助</td><td>版本号：2000</td></tr>
<tr><td>严重程度：次要</td><td>优先级：低</td><td>状态：新建</td></tr>
<tr><td colspan="3">测试人员：×××</td></tr>
<tr><td colspan="3">分配给：</td></tr>
<tr><td colspan="3">日期：2010-08-01</td></tr>
<tr><td colspan="2">硬件平台：
CPU-P4 2.0G
RAM-521M</td><td>操作系统：
Windows 2000 Professional SP4</td></tr>
<tr><td colspan="3">缺陷概述：“关于”对话框的界面中英文混合</td></tr>
<tr><td colspan="3">详细描述：
1. 打开××字处理软件，执行“帮助”→“关于”菜单命令
2. 在“关于”对话框中存在中英文混合的现象：3 个英文单词</td></tr>
<tr><td colspan="3">附件：可附图</td></tr>
<tr><td>解决信息</td><td colspan="2">验证信息</td></tr>
<tr><td>解决人：××程序员</td><td colspan="2">验证人：××测试员</td></tr>
<tr><td>解决版本：1.0</td><td colspan="2">验证版本：2.0</td></tr>
<tr><td>解决时间：2010-08-01</td><td colspan="2">验证时间：2010-08-02</td></tr>
<tr><td>备注：已解决</td><td colspan="2">备注：已通过验证</td></tr>
<tr><td>……</td><td colspan="2">……</td></tr>
</table>

下面介绍提交缺陷报告的一些注意事项。

1)确保重现 BUG

如果无法重现测试人员所提交的 BUG,将会影响开发人员的开发效率,也会影响测试人员自身的声誉,因此在提交 BUG 之前一定要确保 BUG 能够重现。对于严重程度较高的 BUG,一般要重复测试两次以上;对于即时产生的 BUG,要在其他机器上测试一下,看看是否是自己机器的原因。

2)要用最少且必要的步骤描述 BUG

使用最少且必要的步骤,是为了节省开发人员定位问题的时间,例如:

(1)运行客户端。

(2)为输入新的条目查询数据库。

(3)打开一个浏览器。

(4)在 www. yahoo. com 上浏览新闻。

(5)关闭浏览器。

(6)选择一个条目。

(7)把种类从“鲜花”更改为“水果”。

(8)使数据库服务器脱机。

(9)尝试保存记录。

(10)收到一个超时的错误。

(11)退出客户端。

这个缺陷报告的详细描述中共有 11 个步骤,其中步骤(3)、(4)、(5)、(7)是多余的,应该去掉。

3)简洁、准确、完整

测试人员在提交缺陷报告的时候,要站在开发人员的角度上思考问题,要确保开发人员拿到缺陷报告后马上就能够定位问题,而不会产生理解上的歧义。下面是几个基本的要求:

- 缺陷概述:简洁、准确、完整,揭示错误实质,最好用陈述句,一般不超过 15 个字。
- 详细描述:简洁、准确、完整,保证快速准确地描述错误。“简洁”即没有多余的步骤,“准确”即步骤正确,“完整”即没有缺漏。
- 尽量使用业界惯用的表达术语和表达方法。
- 检查拼写和语法错误。

4)一个 BUG 一个报告

有的测试人员喜欢在一个缺陷报告里面提交多个 BUG,这种习惯不值得提倡,原因有以下两点:

- 不便于分配 BUG。如一个缺陷报告里面有两个 BUG,分别属于甲和乙两个开发人员,那么应将该报告提交给甲还是乙?
- 不便于回归测试。测试人员在回归测试的时候需要关闭 BUG,如一个缺陷报告里有两个 BUG,BUG1 已经解决,BUG2 还没有解决,那么缺陷报告是否应该关闭?

总之,尽量使一个缺陷报告只包含一个 BUG。

1.3.3　BUG 的处理流程

很多公司招聘测试工程师经常考的一道题目就是画出 BUG 的处理流程,以考查应聘者

是否具有一定的缺陷管理经验。

实际上,不同公司的 BUG 处理流程一般是不同的,同一公司不同项目的 BUG 处理流程也不尽相同。下面给出某公司 BUG 处理的大致步骤,读者可以结合自己的实际情况灵活修改。

1)发布 BUG

主流程:

(1)测试人员测试功能,并发现 BUG。

(2)测试人员将 BUG 录入到 VSTS 对应的团队项目下,设置 BUG 状态为“已建议”,并认真填写发现环境、发现途径、症状和重现步骤,如有必要还可添加 BUG 的截图。其中重现步骤必须详细到说明直接引发 BUG 的每个用户操作(被操作的控件、被选择的数据等)。

(3)测试人员确定 BUG 修改人,通常情况下,BUG 所属功能的开发人员就是 BUG 修改人。

(4)测试人员在 VSTS 中指派 BUG 的修改人。

(5)如 BUG 修改人在现场实施,测试人员将 BUG 记录在“质量问题反馈表”中,并在每天下班前用邮件方式通知该修改人,如必要可用其他方式及时通知。

2)接收 BUG

主流程:

(1)开发人员在每天上班的第一时间登录 VSTS 或接收邮件,查看需要修改的 BUG。

(2)开发人员确定 BUG 的处理顺序,通常情况下,优先处理高严重级别和高优先级别的 BUG。

(3)开发人员确定能够在规定时间内执行修改的 BUG,将 BUG 状态改为“活动”。

3)修改 BUG

主流程:

(1)开发人员确定在开发环境下可重现 BUG。

(2)开发人员修改 BUG。

(3)开发人员修改 BUG 成功,并测试修改后的功能,在确定没有因修改而产生其他 BUG 之后,将 BUG 状态改为“已解决”,原因改为“已修复”。如在修改过程中遇他人大力协助,可在建议的修改方案中进行说明。

(4)位于公司的开发人员在每天的 16:30 之前将修改成功的代码嵌入 VSTS。位于现场的开发人员在每天的 16:00 之前将修正过的“质量问题反馈表”和修改成功的源代码发给相关测试人员。

(5)位于公司的测试人员在 16:05 接收现场开发人员发回的“质量问题反馈表”和源代码,并在 16:30 之前更新 VSTS 中相关的 BUG 信息和源代码。

1.4 测 试 用 例

1.4.1 测试用例的定义

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。

统一软件开发过程(rational unified process，RUP)认为测试用例是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 Std 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，如执行一条特定的程序路径来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地揭示软件缺陷。

测试用例可以用一个简单的公式来表示，即：

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、硬件环境和相关数据，有时还包括网络环境。

1.4.2　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，软件设计有概要设计(结构设计/架构设计)和详细设计两种，所以对于测试用例的质量标准，也应分为两个层次来考虑高层次和低层次。高层次满足某一个测试目标或测试任务，从整体测试用例看，需衡量一组测试用例的结构、设计思路和覆盖率等指标。低层次从单个测试用例看，需衡量测试用例描述的规范性、可理解性和可维护性等指标。

1. 高层次标准

高层次(high-level)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的需求，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，也就是说，每个测试点都有相应数量的测试用例来覆盖，可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，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应区别对待。

(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.低层次标准

低层次(low-level)标准考查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,也就越容易被理解和执行。执行效率越高就说明测试用例越好。

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,通过一个测试用例只能发现 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 “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”等样式的描述就比较好。

(5)操作步骤的准确性。按照步骤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”,这实际上是不可验证的。要使这个期望结果具有可验证性,就应该这样描述所期望的结果——“‘退出(logout)’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,测试用例就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,非常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,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,不断地提升测试的质量和效率。

1.4.3 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。例如,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了不同的测试用例。

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。这是指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。这是指对同样的测试用例,系统的执行结果应当相同。测试结果确保缺陷的重现,为缺陷的快速修复奠定基础。

而以上 3 条原则中,最难保证的就是测试用例的代表性,这也是设计测试用例时最为关键的内容,即如何确定测试用例中输入的数据集合。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。例如,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。需要注意的是,对于无效输入有两种情况:一是满足所有输入的数据类型要求,但不在有效取值范围内;二是部分输入不满足输入的数据类型要求。更严格地来讲,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。

1.4.4　测试用例模板

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,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 BUG 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。

如图 1-2 所示的模板就是一个测试用例模板。

项目/软件	技术出口合同网络申领系统		项目版本	1. 0. 25		
功能模块名	Login		编 制 人	×××		
用例编号	TC-TEP_Login_1		测试执行时间	2010. 10. 12		
相关用例	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					
预置条件	无		特殊规程说明	对数据库访问权限		
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，单击“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，单击“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	

图 1-2 测试用例模板

1.5 测 试 环 境

1.5.1 测试环境的定义

软件的测试环境就是软件运行的平台，即软件、硬件、网络和历史数据的集合，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不一样，所以在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。

以上就是搭建软件测试环境所需要考虑的 4 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。

1.5.2　测试环境的要素

良好的测试环境应具备以下 3 个要素。

1）良好的测试模型

良好的测试模型有助于高效地发现缺陷，它并不仅仅包含一系列测试方法，更重要的是，它是在长期实践中积累下来的一些历史数据，包括有关某类软件的缺陷分类规律、有关项目小组历次测试的过程数据等。这些历史数据可以相应地反映出针对某类软件的测试关注点、项目小组的开发与测试效率等内容，有助于提高后续类似产品的开发与测试的效率。

2）多样化的系统配置

测试环境在很大程度上应该是用户的真实使用环境，或至少应搭建模拟的使用环境，使之尽量逼近软件的真实运行环境。为此，应测试在各种系统配置条件下软件的运行情况，特别是通用型软件系统的测试。

3）熟练使用工具的测试人员

在系统测试尤其是性能测试环节，通常需要有自动化测试工具的支持，否则将无法模拟或再现系统运行环境；但若仅有测试工具，没有熟悉如何使用工具的测试人员，或缺乏了解被测软件系统特性的测试人员，是无法发挥自动化测试工具的巨大优势的。

1.5.3　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，需要搭建的环境主要用于被测软件系统（包括 Web 应用）的测试。

规划测试环境的第一步应明确如下问题：

（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（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）和数据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。

（4）测试机所需支撑软件环境中各软件的名称、版本，必要时应明确相关补丁的版本。

（5）用于对被测软件系统的服务器环境和测试管理服务器环境进行备份的专用计算机（该环节是可选的）。这对安全性测试、可恢复性测试等会导致重大软件缺陷的测试类型是非常重要的，测试后若出现重大缺陷，应能利用备份来恢复测试前的原始环境。

(6)测试所需的网络环境。

(7)执行测试工作所需的一些辅助软件。例如,文档编写工具、测试管理系统、性能测试工具、缺陷管理系统等,应明确这些软件的名称、版本、授权证书数量和可能需要的相关补丁的版本。对于性能测试工具,还需要重点留意是否支持被测软件系统所用的协议。

(8)为执行测试用例所需初始化的各项数据。对性能测试而言,还需重点留意执行测试用例之前应满足的历史数据量,以及在测试过程中受到影响的数据的恢复问题。

明确以上问题之后,第二步是确定哪些条件可以得到满足,哪些条件需要其他部门来协调、采购或支援。

最后将上述问题整理为检查表,为每个问题制定负责人,在搭建测试环境的过程中,对照检查表来逐步完成每个问题的设置和检查,填写相关内容,最终形成的文档就作为测试环境的配置说明文档。若时间或其他条件允许,还需要做好应急预案,尽量保证在环境失效时不会对正常工作产生太大的影响。

1.5.4 测试环境的维护和管理

搭建好测试环境之后,通常还需要不断地调整环境。例如,软件新版本的发布、需求的变化等,都有可能对测试环境产生影响。为此,应考虑如下问题。

1)设置测试环境管理员

每个测试小组都应配备一名专门的测试环境管理员,其职责如下:

(1)搭建测试环境,包括安装和配置支撑软件环境涉及的所有必需软件,编写相关安装、配置手册。

(2)详细记录构成测试环境的各台机器的硬件配置、IP 地址、端口配置、机器用途和当前网络环境情况。

(3)部署被测软件系统,编写发布文档。

(4)执行和记录测试环境的各项变更情况。

(5)备份和恢复测试环境。

(6)有效地管理支撑软件环境所涉及的所有用户名、密码和权限。

(7)当测试小组内的多名成员都需要占用服务器,且相互存在冲突的时候,负责分配和管理服务器时间。

2)明确测试环境管理所需的文档

为了对测试环境进行管理,需要以下文档:

(1)各台机器上支撑软件环境中各项软件的安装配置手册。

(2)各台机器的硬件环境文档。

(3)被测软件系统的发布手册,特别要注意记录数据库表的创建、数据导入等内容。

(4)测试环境的备份和恢复方法手册。

(5)用户权限管理文档。

对于每个文档,都应详细记录各项信息的变更历史。

3)管理测试环境的访问权限

应为每个访问测试环境的测试人员和开发人员设置单独的用户名,根据工作需要设置各自的访问权限,以避免误操作从而破坏测试环境,具体原则如下:

(1)测试环境管理员统一管理访问支撑软件环境和被测软件系统涉及的所有用户名、密码及权限。

(2)测试环境管理员拥有全部的权限。

(3)对于开发人员,仅给予对被测软件系统的访问权限,如有特殊要求,可给予测试环境其他部分的只读权限。

(4)对于普通测试员,不给予删除权限。

4)管理测试环境的变更

对于变更,最重要的是能够对每次变更进行追溯和控制,基本的原则如下:

(1)由开发人员或测试人员提出书面申请变更测试环境,由测试环境管理员负责执行,不接受任何非正式的变更申请(如口头申请)。

(2)对测试环境的任何变更都应该记入相应的文档。

(3)与每次变更相关的变更申请文档、软件、脚本等,一律保留原始备份,并作为配置项进行管理。

(4)开发人员将整个系统(包括数据库、应用层、客户端等)打包为可直接发布的格式,由测试环境管理员负责实施发布。测试环境管理员不接受任何不完整的版本发布申请。

5)备份和恢复测试环境

测试环境必须可恢复,因为缺陷必须重现,无法恢复测试环境就意味着无法执行原有的测试用例,缺陷重现更加无从谈起,更不要说修复了。一般应在以下情况下考虑对测试环境进行备份:

(1)当测试环境(尤其是软件环境)发生重大变动时需对其备份,主要是在安装或卸载测试中会碰到这样的情况。

(2)每次发布软件新版本时,应做好当前版本的数据备份工作。

(3)性能测试之前,应做好数据库备份或充分准备好数据的恢复方案。

1.6　软件测试职业

1.6.1　国内外软件测试的现状

在软件行业比较发达的国家,特别是美国,软件测试已经发展成为一个独立的产业,主要体现在以下几个方面:

(1)软件测试在软件公司中占有重要的地位。比尔·盖茨曾在麻省理工学院的一次演讲中说:“在微软,一个典型的开发项目组中的测试工程师要比编码工程师多得多,可以说我们花费在测试上的时间要比花费在编码上的时间多得多。”

(2)软件测试理论研究蓬勃发展。每年举办各种各样的测试技术年会,发表大量的软件测试研究论文,引领软件测试理论研究的国际潮流。

(3)软件测试市场繁荣。美国有一些公司专门开发软件测试标准与测试工具,MI、Compuware、Macabe、Rational 等都是著名的软件测试工具提供商,它们出品的测试工具已经占领了国际市场,目前的主流测试工具大部分都是它们的产品,可见国外的软件测试已经形成

了较大的产业规模。

中国的软件测试技术研究起步于“六五”期间，主要是随着软件工程的研究而逐步发展起来的，由于起步较晚，与国际先进水平相比差距较大。直到 1990 年，成立了国家级的中国软件评测中心，软件测试服务才逐步开展起来。因此，我国无论是在软件测试理论研究还是测试实践上，和国外发达国家相比都有不小的差距，主要体现在对软件产品化测试的技术研究还比较贫乏、从业人员较少、测试服务没有形成足够的规模等方面。但是，随着我国软件产业的蓬勃发展以及对软件质量的逐渐重视，软件测试也越来越被人们所看重，软件测试正逐步成为一个新兴的产业。

1.6.2 软件测试人员结构

软件测试工程师是软件行业中一种既年轻又古老的职业，进入 21 世纪以来，随着中国加入 WTO，从事这项职业的人也越来越多。一个公司在组建一支测试队伍的时候如何分配人员结构，从而使公司软件测试水平得到提高，是一个备受关注的问题。下面介绍软件测试人员的基本结构。

1）测试经理

测试经理主要负责测试队伍的内部管理以及与其他外部人员、客户的交流，具体来说，主要包括进度管理、风险管理、资金管理、人力资源管理、交流管理等。测试经理需要具有项目经理的知识和技能，同时测试工作开始前测试经理需要撰写测试计划书，测试结束需要书写测试总结报告。

2）测试文档审核师

测试文档审核师主要负责前置测试，包括对在需求期与设计期间产生的文档进行审核，比如业务建模书、需求规格说明书、概要设计书、详细设计书等。审核需要撰写审核报告。当文档确定后，需要整理文档报告，并反映给测试设计师。

3）测试设计师

测试设计师主要根据需求分析期与设计期产生的文档来设计各个测试阶段的测试用例。一般来说，测试文档审核师、测试设计师可以由一组相同的人员来完成。

4）测试工程师

测试工程师按照测试用例来完成测试工作。

1.6.3 软件测试人员的素质要求

1）工作态度好，主动性高

工作态度如何，是评价一个软件测试人员的很重要的方面。一个技术能力强的软件测试人员如果没有好的工作态度，在测试团队中不但不能对测试工作起到推动作用，反而可能起到阻碍作用；而一个工作态度好的测试人员，哪怕他的技术水平不高，人也不聪明，但对自己的工作认真负责，也会对测试工作起到很大的促进作用。

2）认真、细心、耐心

软件测试工作是一项烦琐的工作，需要测试人员具备认真、细心、耐心等素质。

有一句话说：细节决定成败。这句话格外适用于软件测试人员。软件测试，简单来说，就是找缺陷以保证产品质量。在一轮又一轮、成千上万的测试用例中发现尽可能多的缺陷，

认真、细心、耐心是一名优秀的测试人员必备的素质要求。

3)学习理解能力强,善于学习总结

不断地学习新技术,不断总结在实际工作中遇到的问题以及解决的方法,并把它们整理归纳,这是一名软件测试人员提高自己技术水平的最好方法。

4)软件测试理论的掌握

开发工具在变,软件测试工具在变,被测试的系统在变……一切都在变。作为一个测试人员,应该怎么去变呢?测试的类型有很多种,有软硬件测试,有黑白灰盒测试,有功能/系统/压力/Beta(即 β)测试等,但不管系统用的是什么测试,基本步骤是不变的。首先都需要开发人员提供比较好的需求文档、概要/详细设计文档。需求文档是制定测试需求的标准,也是判断系统是否存在问题的标准;概要/详细设计文档是制作测试用例的依据,划分等价类、边界值等基本测试方法都需要这些文档的支持。当然,每一种不同类型的测试都有其特殊的地方,如蓝牙测试就需要对其协议/通信理论(系统环境)有一定的了解,也就是说,好的测试人员必须能够熟练掌握测试理论,做到举一反三。

5)熟悉开发工具和平台

不了解开发平台是无法做单元测试的,也无法做好性能测试,更无法扩展自己的软件测试知识面,了解测试深度。

6)掌握测试工具

测试人员必须至少熟练掌握 1～2 种测试工具。

习　题　1

1. 软件开发与软件测试有何关系?

2. “BUG 就是程序运行时产生的错误”这句话对吗?为什么?

3. 怎样的测试用例才算是一个良好的测试用例?

4. 什么是软件测试环境?