21世纪高等学校应用型教材

RUANJIAN CESHI JISHU

软件测试技术

■ 谢进军　王 岩　主编

RUANJIAN CESHI JISHU

中国计量出版社
CHINA METROLOGY PUBLISHING HOUSE

21世纪高等学校应用型教材

软件测试技术

谢进军　　王　岩　主编

中国计量出版社

图书在版编目（CIP）数据

软件测试技术/谢进军，王岩主编．—北京：中国计量出版社，2008.8
21世纪高等学校应用型教材
ISBN 978-7-5026-2787-4

Ⅰ．软…　Ⅱ．①谢…　②王…　Ⅲ．软件—测试—高等学校—教材　Ⅳ．TP311.5

中国版本图书馆CIP数据核字（2008）第001956号

内 容 提 要

本书详细介绍了软件开发技术、软件缺陷的种类、测试用例、配置测试环境、软件测试的问题跟踪系统、软件测试工具、软件评价算法、软件测试文档等。

本书借鉴国内外同类书籍、全面系统地阐述了软件测试技术中所涉及的技术、工具和方法。内容翔实、新颖，概念清晰，通俗易懂，实用性强。本书不仅可作为高等院校、高职高专机电类专业教材，亦可作为相关岗位培训教材，还可供有关技术人员阅读参考。

中国计量出版社 出版
地　址　北京和平里西街甲2号（邮编100013）
电　话　（010）64275360
网　址　http://www.zgjl.com.cn
发　行　新华书店北京发行所
印　刷　北京市密东印刷有限公司
开　本　787mm×1092mm　1/16
印　张　15
字　数　356千字
版　次　2008年9月第1版　2008年9月第1次印刷
印　数　1—3 000
定　价　27.00元

本书编委会

主　编　谢进军　王　岩

副主编　郑凤仁　刘寅生　李　莹　刘明信
郑昆岩　王淑艳　梁　爽

编　委　田　丹　刘申菊　杨　柯　徐香坤
李冬明　王　婷　刘树民　苏　明
于景河　栾凤慧　郑雅琳　郑凤敏
曹雨顺　刘　辉　陈久俊　刘继承
韩　松

前言

• FOREWORD •

信息技术的飞速发展，使软件产品应用到社会的各个领域，软件产品的质量自然成为人们共同关注的焦点。不论软件的生产者还是软件的使用者，均生存在竞争的环境中，软件开发商为了占有市场，必须把产品质量作为企业的重要目标之一，以免在激烈的竞争中被淘汰出局。

有错是软件的属性，因为软件是由人来完成的，所有由人做的工作都不会是完美无缺的。问题在于我们如何去避免错误的产生和消除已经产生的错误，使程序中的错误密度达到尽可能低的程度。采用新的语言、先进的开发方式、完善的开发过程，可以减少错误的引入，但是不可能完全杜绝软件中的错误，这些引入的错误需要测试来找出，软件中的错误密度也需要测试来进行估计。测试是所有工程学科的基本组成单元，是软件开发的重要部分。

软件测试是软件质量保证的关键步骤。软件测试研究的结果表明：软件中存在的问题发现越早，其软件开发费用就越低；在编码后修改软件缺陷的成本是编码前的10倍，在产品交付后修改软件缺陷的成本是交付前的10倍；软件质量越高，软件发布后的维护费用越低。另据对国际著名IT企业的统计，它们的软件测试费用占整个软件工程所有研发费用的50%以上。统计表明，在典型的软件开发项目中，软件测试工作量往往占软件开发总工作量的40%以上。而在软件开发的总成本中，用在测试上的开销要占30%到50%。如果把维护阶段也考虑在内，讨论整个软件生存期时，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，其中必定还包含有许多测试工作。因此，测试对于软件生产来说是必需的，问题是我们应该思考“采用什么方法、如何安排测试”，才能够使软件错误降到最低程度，使软件质量得以保证。

全书全分9章，涵盖了软件测试技术和方法所涉及的各方面内容，包括软件测试技术、测试环境的建立、软件测试的问题跟踪系统等，既有理论方法，又有实践经验。

第 1 章为概述，介绍软件的相关定义、软件测试概述以及软件测试管理流程等内容。

第 2 章主要介绍软件开发过程和在软件开发过程中所采用的过程模型，结合过程模型来阐述软件测试的地位，还介绍了软件工程技术、面向对象程序设计基础和能力成熟度模型 CMM 等。并力图从案例中给读者一些启发。

第 3 章介绍“软件缺陷”这个重要概念，以 BUG 为出发点引出造成软件缺陷(bug) 的原因、软件缺陷的种类，以及软件测试的分类、阶段和过程。

第 4 章主要介绍的是软件测试技术，重点介绍软件测试技术、策略和测试用例的内涵、种类及设计方法，是本书的重点章节。

第 5 章软件测试环境的建立，介绍一个标准的、规范的测试环境是如何建立和配置起来的，以及如何做好维护，满足测试对环境的严格要求。

第 6 章软件测试的问题跟踪系统，主要介绍什么是问题跟踪系统、问题跟踪系统的功能、运用及建立等。

第 7 章介绍软件测试工具，包括 Compuware NuMega 侦错软件、GUI 接口自动化测试、Network Sniffer Pro、Sysinternals 所提供的工具等。

第 8 章软件评价算法，介绍 COCOMO 开发进度估算法、EQF 预估进度准确度、软件缺陷比率的估算。

第 9 章介绍软件测试的相关文档，主要是测试计划的规范与要求。

由于水平和时间的限制，书中不可避免会出现一些错误，请各界同仁不吝赐教。

编　者

2008 年 7 月

目 录

• CONTENTS •

第一章 概 述 …… (1)

第一节 软 件 …… (1)
第二节 软件测试概述 …… (5)
第三节 软件测试管理流程 …… (13)
思考题与习题 …… (15)

第二章 软件开发技术 …… (17)

第一节 软件过程 …… (17)
第二节 案例说明 …… (34)
第三节 软件工程技术 …… (37)
第四节 面向对象程序设计基础 …… (41)
第五节 能力成熟度模型 CMM …… (52)
思考题与习题 …… (58)

第三章 软件缺陷的种类 …… (59)

第一节 Bug 的历史 …… (59)
第二节 造成软件缺陷的原因 …… (60)
第三节 缺陷的种类 …… (65)
思考题与习题 …… (70)

第四章 测试用例 …… (71)

第一节 测试用例的概念 …… (71)
第二节 为什么软件测试需要测试用例 …… (72)
第三节 测试用例的种类 …… (73)
第四节 测试用例设计技巧 …… (105)

第五节　软件测试技术 …… (107)
第六节　测试策略 …… (124)
第七节　调试 …… (132)
思考题与习题 …… (135)

第五章　配置测试环境 …… (139)

第一节　测试环境的快速变迁 …… (139)
第二节　配置测试环境的困难点 …… (140)
第三节　如何配置测试环境 …… (142)
第四节　测试环境配置需求清单 …… (147)
第五节　测试环境与外界真实环境 …… (148)
第六节　个案讨论 …… (149)
思考题与习题 …… (151)

第六章　软件测试的问题跟踪系统 …… (152)

第一节　实施目的 …… (152)
第二节　问题的生命周期 (Bug Lifecycle) …… (154)
第三节　设置问题的等级 (Bug Priority and Bug Severity) …… (155)
第四节　系统的基本功能 (Basic Functions) …… (156)
第五节　如何运用问题跟踪系统 …… (160)
第六节　购买还是自行建置 …… (164)
思考题与习题 …… (167)

第七章　软件测试工具 …… (168)

第一节　使用软件测试工具的目的 …… (168)
第二节　测试工具的种类 …… (170)
第三节　Compuware NuMega 侦错软件 …… (172)
第四节　GUI 接口自动化测试 …… (177)
第五节　Network Snifferr Pro …… (183)
第六节　Sysinternals 所提供的工具 …… (189)
第七节　其他测试工具 …… (193)
思考题与习题 …… (196)

第八章　软件评价算法 …… (197)

第一节　软件可靠性 …… (197)

第二节　COCOMO 开发进度估算法 …… (200)

第三节　EQF 预估进度准确度 …… (202)

第四节　软件缺陷比率的估算 …… (204)

思考题与习题 …… (206)

第九章　软件测试文档 …… (208)

第一节　软件测试计划 …… (208)

第二节　软件测试所需文件 …… (216)

思考题与习题 …… (227)

参考文献 …… (228)

第一章 概 述

计算机系统的发展与微电子技术的进步息息相关。自 1946 年计算机诞生以来，计算机系统已经经历了电子管、晶体管、集成电路、大规模集成电路、超大规模集成电路等多个不同的发展时期。由于微电子学技术的进步，计算机硬件性能/价格比平均每十年提高二个数量级，而且质量稳步提高。与此同时，正在运行使用着的计算机软件的数量正以惊人的速度急剧膨胀，计算机软件成本在逐年上升，而且质量没有可靠的保证，软件开发的生产率也远远跟不上计算机应用的要求，软件已经成为限制计算机系统发展的关键因素。

第一节 软件

一、软件的定义

软件是计算机系统中与硬件相互依存的部分，它是包括程序、数据及相关文档的完整集合。其中，程序是按事先设计的功能和性能要求执行的指令序列；数据是程序所处理信息的数据结构；文档是与程序开发、维护和使用相关的各种图文资料。

二、软件的特点及最新发展

为全面、正确地理解计算机系统及其软件，我们必须了解软件的以下特点。

1. 抽象性

软件是一种逻辑实体，而不是具体的物理实体。这种抽象性是软件与硬件的根本区别。软件一般寄生在诸如纸、内存储器、磁带、磁盘或光盘等载体上，我们无法观察到它的具体形态，而必须通过对它的运行来分析了解它的功能和特征。

2. 无明显的制造过程

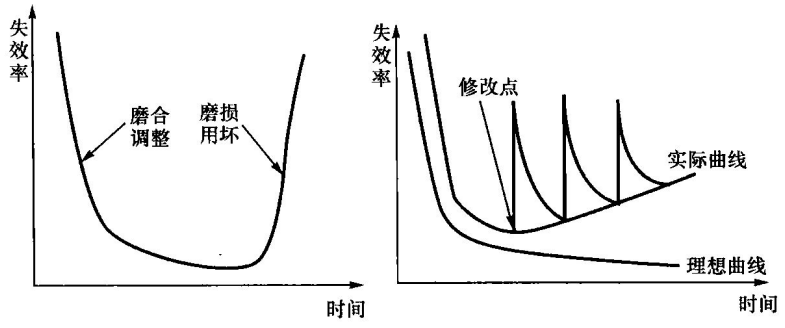
软件的生产与其他硬件的生产不同，它没有明显的制造过程。在硬件的制造过程中必须对每一个制造环节都进行质量控制，以保证整个硬件产品的质量，并且每一个硬件都几乎付出与样品一样的生产资料成本。而软件是将人类的知识和技术转化成产品，软件产品的开发成本几乎全部用在样品的开发设计上，其制造过程则非常简单，人们可以用很低的成本进行软件产品的复制，因此，也产生了软件产品的保护问题。除国家在法律上采取有力措施之外，开发者在技术上也采取了各种措施，防止对软件产品的随意复制。

3. 无磨损、老化的问题

在软件的运行和使用期间，没有像硬件那样的磨损、老化问题。任何机械、电子设备在运行和使用的过程中，其失效率大致遵循如图 1—1（a）（硬件失效率曲线）中所示的 U 形曲线（即浴盆曲线）。软件的情况则与此不同，它不存在磨损和老化的问题，然而它却存在退化的问题，设计人员必须不断地修改软件，如图 1—1（b）（软件失效率曲线）所示。

4. 对硬件系统的依赖性

软件的开发和运行往往受到计算机系统的限制，对计算机系统有着不同程度的依赖性。为了减少这种依赖性，在软件开发中提出了软件的可移植问题。



(a) 硬件失效率曲线　　(b) 软件失效率曲线

图 1—1　硬件、软件失效率曲线

5. 软件开发尚未完全摆脱手工艺的方式

目前，软件开发尚未完全摆脱手工艺的方式。

6. 复杂性

软件本身是复杂的。软件的复杂性可能来自它所反映的实际问题的复杂性，也可能来自程序逻辑结构的复杂性。

7. 成本昂贵

软件的研制工作需要投入大量的、复杂的、高强度的脑力劳动，它投入的成本是较高的。

8. 社会性

相当多的软件工作涉及各种社会因素，许多软件的开发和运行涉及机构设置、体制运作及管理方式等问题，甚至涉及人们的观念和心理，这些因素直接影响到软件项目的成败。

自 20 世纪 40 年代世界上第一台计算机问世以来，软件经历了程序设计、程序系统及软件工程 3 个阶段的发展。程序设计阶段（20 世纪 50 年代至 60 年代）的软件指的是程序，程序的开发采用个体工作的方式，开发工作主要依赖于开发人员的个人技能和程序设计的技巧，软件的质量得不到保证，缺乏与程序有关的文档；程序系统阶段（20 世纪 60 年代至 70 年代）的软件是指程序和说明书，软件开发采用开发小组工作的方式，开发工作主要依赖开发小组的水平，文档资料的不齐全给软件维护带来了很大的难度，软件技术的发展不能满足需要；软件工程阶段（20 世纪 70 年代以后）的软件则是指程序、文档、数据，由开发小组及大中型软件开发机构承担开发软件的任务，开发工作主要依赖于整个机构的管理水平，采用多种开发技术。

目前，在很多的应用领域，人们开始采用面向对象的软件开发技术。专家系统、人工智能软件开始走向实际应用。软件技术呈现国际化、网络化、服务化等多种发展趋势。互联网作为 20 世纪最重要的科技成果之一，给人类生活和经济发展都带来了深远的影响，它所展现出的勃勃商机，吸引了众多厂商围绕互联网开发软件，与分布计算、网络和互联网相关的软件技术成为软件领域的主要技术热点。此外，自由软件潮流、智能化、简易化、多样化等

趋势正极大地拓展软件产业的发展空间，派生出许多具有成长潜力的新兴领域。

三、软件的分类

20世纪40年代以来，尽管人们开发了大量的软件，积累了丰富的软件资源并使之广泛应用于各个领域，但软件的品种、质量和价格方面仍然满足不了人们日益增长的需要。随着软件复杂性和交互性的增加，我们难以对目前应用着的软件进行一个标准化的分类，这里只是简单地介绍计算机软件在计算机系统、实时系统、商业管理、科学和工程计算、嵌入式系统和人工智能等方面的应用。

1. 系统软件

系统软件是指能与计算机硬件系统紧密配合，使计算机系统的各个部件、相关软件和数据协调高效地工作的软件。系统软件是计算机系统的重要组成部分，它支持应用软件的开发和运行。系统软件包括：操作系统、网络软件、编译程序、数据库管理程序、文件编辑系统、系统检查与诊断软件等。

2. 实时软件

监视、分析和控制现实世界中发生的事件，能以足够快的速度对输入信息进行处理并在规定的时间内作出反应的软件，称之为实时软件。实时软件包括4个组成部分：数据采集器（负责从外部环境中获取和格式化信息）、分析器（负责将信息转换成应用所需要的形式）、输出/控件器（负责响应外部环境）、管理器（负责协调系统各个部件工作，使系统能保持在一个可接受的响应时间内给实时响应）。实时系统必须在严格的时间范围内响应，因此，实时软件和计算机系统必须有很高的可靠性和安全性。

3. 商业软件

商业软件可以访问一个或多个商业信息的大型数据库，将已有的数据重新构造，变化成一种能够辅助商业操作和管理决策的形式。商业信息处理是当今最大的软件应用领域，典型的商业软件有银行储蓄软件、电子商务软件、仓库管理软件和ERP软件等。

4. 科学和工程计算软件

科学和工程计算软件的特征是“数值分析算法”，它主要用于科学和工程的计算。如天气预报、弹道计算、石油勘探、地震数据处理、计算机系统仿真和计算机辅助设计等。

5. 嵌入式软件

嵌入式计算机系统将计算机嵌入在某一系统之中，使之成为该系统的重要组成部分，控制该系统运行，进而实现一个特定的物理过程。用于嵌入式计算机系统的软件称为嵌入式软件。大型的嵌入式计算机系统软件可用于航空航天系统、指挥控制系统、武器系统等；小型的嵌入式计算机系统软件可用于工业的智能化产品之中，这时嵌入式软件驻留在只读存储器内，为该产品提供各种控制功能和仪表的数字或图形显示功能等。如汽车的刹车控制、空调机、洗衣机的自动控制等。

6. 人工智能软件

人工智能软件利用非数值算法去解决复杂的问题，一般说来，这些问题都不能通过计算或直接分析而得到答案。迄今为止，在专家系统、模式识别、自然语言理解、人工神经网络、程序验证、自动程序设计、机器人学等领域开发了许多人工智能应用软件，用于诊断疾病、产品检测、自动定理证明、图像和语音的自动识别、语言翻译等。

四、软件危机

20世纪60年代中期至20世纪70年代中期，“软件危机”一词在计算机界广为流传。这个时期的一个重要特征是出现了“软件作坊”，广泛使用产品软件。“软件作坊”基本上仍然沿用早期形成的个体化软件开发方法。同时，随着计算机应用的日益普及，软件数量急剧膨胀，在程序运行时发现有错误必须及时改正；用户有了新的需求时必须相应地修改程序；硬件或操作系统更新时需要修改程序以适应新的环境。上述的种种维护工作，以令人吃惊的比例耗费资源。更严重的是，许多程序的个体化特性使得它们最终成为不可维护的系统，于是“软件危机”开始出现了。1968年，在前联邦德国召开的北大西洋公约组织的国际会议上，计算机科学家们讨论了软件危机的问题。在这次会议上正式提出并使用了“软件工程”这个名词，一门新兴的学科就此诞生了。

五、软件危机的原因及解决方法

软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。产生软件危机的原因主要有以下几点。

（1）由于缺乏软件开发的经验和有关软件开发数据的积累，使得开发工作的计划很难制定，以致经常出现超出经费预算，无法遵循进度计划，完成开发的期限一再拖延等情况。

（2）软件需求在开发的初期阶段不够明确，或是未能得到确切的表达。开发工作开始后，软件人员和用户又未能及时交换意见，造成矛盾在开发后期集中暴露。

（3）开发过程没有统一、公认的方法论和规范进行指导，参加开发的人员各行其是。另外，设计和实现过程的资料很难维护。

（4）未能在测试阶段做好充分地检测工作，提交至用户的软件质量差，在运行过程中暴露出大量的问题。

如果这些障碍不能有效地突破，软件的发展是没有出路的。因此，有许多计算机软件专家尝试把其他工程领域中行之有效的工程学知识运用到软件开发工作中来。经过不断的实践总结，最后得出结论：按工程化的原则和方法组织软件开发工作是有效的，是解决软件危机的一个重要方法。由此，软件工程成为了计算机科学技术中的一个新领域，它从管理和技术两方面研究如何更好地开发和维护计算机软件，有效地缓解了软件危机所引发的种种问题。

六、软件工程的概念

软件工程是指应用计算机科学、数学及管理科学等原理，以工程化的原则和方法来解决软件问题，指导计算机软件开发和维护的一门工程学科。

软件工程过程是为了获得好的软件产品，在软件开发工具的支持下，由软件开发者即软件工程师完成的一系列软件工程活动。软件工程过程通常包含以下4种基本活动。

（1）软件需求规格说明：确定被开发软件的功能及性能指标，给出软件运行的约束。

（2）软件开发：开发出满足软件需求规格说明的软件。

（3）软件确认：确认软件能够满足客户提出的要求。

（4）软件维护：为了满足用户对软件提出的新的要求，软件必须在使用中不断的维护，以适应用户。

事实上，软件工程过程是一个软件开发机构针对某类软件产品为自己规定的工作步骤，它应当是科学的、合理的，否则必将影响软件产品的质量。

七、软件工程的原则

软件工程的目的是提高软件生产率，提高软件质量，降低软件成本。为了达到这个目的，在软件的开发过程中必须遵循以下软件工程原则。

1. 抽象

抽取事物最基本的特征和行为，忽略非基本细节。采用分层次抽象，自顶向下、逐层细化的办法控制软件开发过程的复杂性。

2. 信息隐蔽

将模块设计成“黑箱”，实现细节隐藏在模块内部，不让模块的使用者直接访问，这就是所谓信息封装（使用与实现分离）的原则。使用者只能通过模块接口访问模块中封装的数据。

3. 模块化

模块是程序中是逻辑上相对自主的成分，是独立的编程单位，应有良好的接口定义。如C语言程序中的函数过程，C++语言程序中的类。模块化有助于信息隐蔽和抽象，有助于表示复杂的系统。

4. 局部化

在一个物理模块内集中逻辑上相互关联的计算机资源，保证模块之间有松散的耦合，模块内部的较强的内聚，这有助于控制软件的复杂性。

5. 确定性

软件开发过程中所有概念的表达应是确定的、无歧义的、规范的。这样有助于人们在交流时不会产生误解、遗漏，保证整个开发工作的协调一致。

6. 一致性

整个软件系统（包括程序、文档和数据）的各个模块应使用一致的概念、符号和术语；程序内、外部接口应保持一致；软件同硬件、操作系统的接口应保持一致；用于形式化规格说明的公理系统应保持一致。

7. 完备性

软件系统不丢失任何重要成分，可以完全实现系统所要求的功能。为了保证系统的完备性，在软件开发和运行过程中需要严格的技术评审。

8. 可验证性

开发大型的软件时需要对系统自顶向下、逐层分解。系统分解应遵循使系统易于检查、测试、评审的原则，以确保系统的正确性。

使用一致性、完备性和可验证性可以帮助开发者设计一个正确的系统。

第二节　软件测试概述

软件测试是软件质量管理中最实际的行动。软件测试是有组织性、步骤性和计划性的。由于软件测试有不同的种类及项目，为了方便管理，通常会将相同类型的测试项目归纳在一

起，而这个动作就是测试的组织性，如性能测试、功能测试等，这是将测试性质相同的项目组织而形成的。而一般的软件测试会根据所设计的测试用例或条例式测试（Test Cases）逐步进行测试。而按照测试用例所执行的测试行为，就是测试的步骤性。必须参考测试计划才能设计出符合软件需求的测试用例，这个流程就是测试的计划性，如图 1—2 所示。

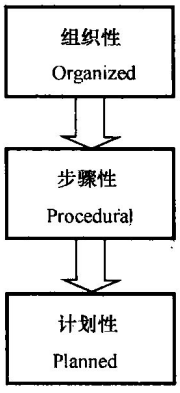


图 1—2　软件测试特性

软件测试的种类可以根据测试形态、测试技术及测试模式这 3 项来进行分类，在这 3 个种类中又可以依照它的特色区分出不同的项目或类别，接下来我们将对这些软件测试种类做一个简单的介绍。

一、测试形态（Testing Types）

以测试形态分类的话，可以分为建构性测试（Construction Testing）、系统测试（System Testing）及专项测试（Special Testing）这 3 大项，如图 1—3 所示。建构性测试属于前置性的测试，它主要偏重于程序端的测试，以确保程序运作正常。系统测试是属于中后期的集成测试，所进行的测试以使用者的观点为主，也就是模仿外界的使用者会如何使用产品。至于专项测试指的是所进行的测试需要花费更多的时间与人力才能完成。

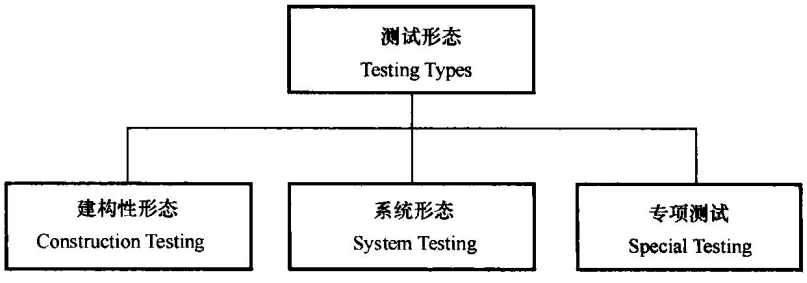


图 1—3　软件测试形态

1. 建构性测试（Construction Testing）

建构性测试又称开发测试（Developing Testing），顾名思义，就是当程序还处于建设阶段时所进行的测试。这类测试通常是由程序开发人员自己来进行的，必要时也要自行开发测

试工具。有些软件公司会指派资历较浅的开发人员进行建构性测试。事实上就组织框架而言，这样的分配对质量帮助有限，因为在找到问题后，资历较浅的开发人员很难去挑战资深的开发人员。另外一点就是，许多的软件公司认为进行过配置测试，就等于已经进行了软件测试，这个误解在国内到处可见。笔者建议由开发人员交替进行配置测试，这样不仅可以避免上述所提到的问题，同时也可避免球员兼裁判的情形发生。在 XP 开发模式中，提倡开发人员交替进行单元测试。

建构性测试有以下 5 种，如图 1—4 所示。

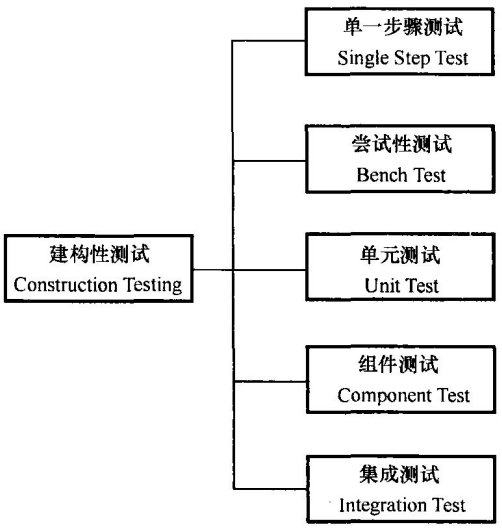


图 1—4　所示建构性测试

（1）单一步骤测试（Single Step Test）：根据程序步骤逐一地进行测试。

（2）尝试性测试（Bench Test）：在开发过程中，为了系统的某一个功能而构建出初步的成品所进行的尝试性测试。

（3）单元测试（Unit Test）：所谓的单元就是将系统切分为细小的个体，单元测试就是针对这些个体来做测试。

（4）组件测试（Component Test）：组件是由一个或多个单元组成的，组件测试的界限会比单元测试大得多。

（5）集成测试（Developer Integration Test）：在软件开发过程中，开发人员是各自开发不同模块的，一旦这些模块编写完成，开发人员必须将这些模块集成起来做一个测试。在这个阶段出错率相当高，而通常出错的原因在于沟通协调上，特别是在大的跨国软件公司中更是如此。

值得注意的是单元测试（Unit Test）。单元测试的目的就是要确认程序的一小部分功能是否能够正确实现，特别是在不同的执行路径上。单元测试目前之所以热门，是因为 XP 开发模式的兴起，在 XP 开发模式中所提倡的测试就是单元测试。根据以往的经验，在面向对象的程序中进行单元测试要比以往困难许多，所幸的是目前已经有许多的单元测试框架（Unit Test Framework）被提出，如 JUnit for Java 或 CppUnit for C＋＋的单元测试框架等，

另外，像 Rational（http：//www. rational. com）为了单元测试也推出了 Test RealTime 的产品。

2. 系统测试（System Testing）

在经过开发人员的集成测试后，可正式将软件编译构建成初步的测试版本（Build Version），之后所要进行的测试就是系统测试。系统测试是针对系统进行测试，这包括所应支持的软件、硬件、操作系统及所应集成的第三方软件（Third Party Software）。系统测试通常称为 QA Testing，测试人员通常为专业的系统测试工程师。

系统测试有以下 6 种，如图 1—5 所示。

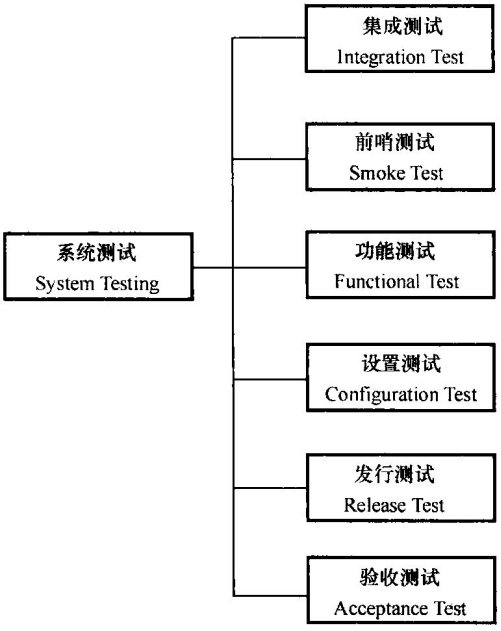


图 1—5　系统测试

（1）集成测试（Integration Test）：专注于系统的稳定度及功能上的测试，特别集成了内外部的子系统及所应支持的第三方软件。

（2）前哨测试（Smoke Test）：每次编译构建的测试版本，必须通过前哨测试决定这个版本是否可以提供给 QA 人员进行系统测试。通常开发人员在修改 Bugs 后有可能引起其他更多的并发问题（Side Effects），通过前哨测试可以推断出问题是在测试版本的第几版发生的。

（3）功能测试（Functional Test）：针对软件在功能上所做的测试，以确保系统达到功能上的要求（Features Requirements）。

（4）设置测试（Configuration Test）：只要是软件都会提供设置功能供使用者做设置，不同的使用者会有不同的设置，不同的设置就会有不同的排列组合测试。

（5）发行测试（Release Test）：这个测试确保软件发表版本可供使用者正常部署、安装和使用，而且功能上必须达到要求。