21世纪软件工程专业规划教材

软件测试

（第2版）

周元哲　编著

清华大学出版社

21世纪软件工程专业规划教材

软件测试

(第2版)

周元哲　编著

清华大学出版社
北京

内 容 简 介

本教材较全面涵盖了当前软件测试领域的专业知识，追溯了软件测试的发展史，反映了当前最新的软件测试理论、标准、技术和工具，展望了软件测试的发展趋势。本教材分为主、辅教材，《软件测试》为主教材，包括软件测试概论、软件测试基本知识、黑盒测试、白盒测试、软件测试流程、性能测试、软件测试自动化和软件测试管理等内容。《软件测试习题解析与实验指导》为辅教材，给出了习题解析，并对软件测试实验进行了指导操作。

适合作为高等院校相关专业软件测试的教材或教学参考书，也可以供从事计算机应用开发的各类技术人员应用参考，或作为全国计算机软件测评师考试、软件技术资格与水平考试的培训资料。

图书在版编目(CIP)数据

软件测试/周元哲编著. —2 版. —北京：清华大学出版社，2017
(21 世纪软件工程专业规划教材)
ISBN 978-7-302-47329-9

Ⅰ. ①软… Ⅱ. ①周… 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(2017)第 115874 号

责任编辑：张　玥
封面设计：常雪影
责任校对：李建庄
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954

印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：9.75　　**字　　数**：237 千字
版　　次：2013 年 9 月第 1 版　2017 年 8 月第 2 版　　**印　　次**：2017 年 8 月第 1 次印刷
印　　数：1～2000
定　　价：29.50 元

产品编号：074447-01

前言

PREFACE

本书第1版自2013年出版以来，深受广大读者的欢迎。经过近几年的教学实践，本书在继承原教材通俗易懂，易于学习的基础上，进行了如下修订。

(1) 软件测试是一门理论与实践紧密联系的课程，直接关系到学生的理论分析能力和综合动手能力的培养。本教材以软件测试技术为主要研究对象，介绍了软件测试的基本理论和基本软件测试工具。

(2) 本教材分为主、辅教材，《软件测试》为主教材，包括软件测试概论、软件测试基本知识、黑盒测试、白盒测试、软件测试流程、性能测试、软件测试自动化、软件测试管理。《软件测试习题解析与实验指导》为辅教材，给出了软件测试习题解析，并对软件测试实验进行了指导操作。

软件测试的先导课为计算机导论、程序设计语言、离散数学、软件工程等课程。软件测试理论较繁杂，让学生在实践中学习理论知识，并用理论知识指导实践，是这本书的写作目的。本书主要使学生掌握软件测试的基本原理、基本方法、基本技术、基本标准和规范，培养学生的合作意识和团队精神，提高学生软件测试的综合能力。

西安邮电大学计算机学院的王曙燕、邓万宇、孟伟君、舒新峰、张昕对本书的编写给予了大力支持，并提出了指导性意见，西北工业大学郑炜、南京大学陈振宇、上海睿亚训软件技术服务公司王磊、韩伟，以及清华大学出版社张玥编辑对本教材的写作大纲、写作风格等提出了很多宝贵的意见。本书在写作过程中参阅了大量中外文专著、教材、论文、报告及网上资料，由于篇幅所限，未能一一列出。在此，向各位作者表示敬意和衷心的感谢。

本书内容精练，文字简洁，结构合理，综合性强，主要面向软件行业初、中级读者，由“入门”起步，侧重“提高”。特别适合作为高等院校相关专业软件测试的教材或教学参考书，也可以供从事计算机应用开发的各类技术人员应用参考，或作为全国计算机软件测评师考试、软件技术资格与水平考试的培训资料。

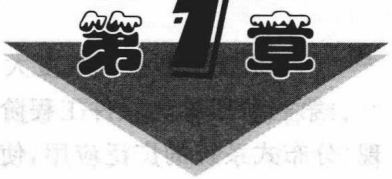
由于作者水平有限，时间紧迫，本书难免有不足之处，诚恳期待读者的批评指正，以使本书日臻完善。我的电子邮箱是 zhouyuanzhe@163.com。

编者

2017年3月

目 录

CONTENTS



软件测试概论

本章介绍了软件的发展历史、软件项目和软件过程，软件测试的由来，软件测试与软件开发的关系，软件缺陷，软件测试行业以及测试认识的误区等内容，为学习后续知识作必要准备。

1.1 软　件

软件是一系列按照特定顺序组织的计算机数据和指令的集合。一般来讲，软件分为编程语言、系统软件、应用软件和介于这两者之间的中间件。其中系统软件为计算机使用提供最基本的功能，但是并不针对某一特定应用领域。而应用软件则恰好相反，根据用户和所服务的领域，不同的应用软件提供不同的功能。

一般认为，软件包括如下内容：

(1) 运行时，能够提供所要求功能和性能的指令或计算机程序集合。

(2) 程序能够满意地处理信息的数据结构。

(3) 描述程序功能需求、程序如何操作和使用所要求的文档。

1.1.1 软件发展史

20 世纪 50 年代初期至 60 年代中期是软件发展的第一阶段，又称为程序设计阶段。此时硬件已经通用化，而软件的生产却是个体化。软件产品为专用软件，规模较小，功能单一，开发者即使用者，软件只有程序，无文档。软件设计在人们的头脑中完成，形成了“软件等于程序”的错误观念。

第二阶段从 20 世纪 60 年代中期至 70 年代末期，称为程序系统阶段。此时多道程序设计技术、多用户系统、人机交互式技术、实时系统和第一代数据库管理系统出现，出现了专门从事软件开发的“软件作坊”，软件被广泛应用，但软件技术和管理水平相对落后，导致“软件危机”的出现。软件危机主要表现在以下几个方面。

(1) 软件项目无法按期完成，超出经费预算，软件质量难以控制。

(2) 开发人员和开发过程之间管理不规范，约定不严密，文档书写不完整，使得软件维护费用高，某些系统甚至无法进行修改。

(3) 缺乏严密有效的质量检测手段，交付给用户的软件质量差，运行中出现许多问题，甚至带来严重的后果。

(4) 系统更新换代难度大。

第三阶段称为软件工程阶段,从20世纪70年代末期至80年代中期,微处理器的出现、分布式系统的广泛应用,使得计算机真正成为大众化的东西。以软件的产品化、系列化、工程化和标准化为特征的软件产业发展起来,软件开发有了可以遵循的软件工程化的设计准则、方法和标准。1968年,北大西洋公约组织的计算机科学家在联邦德国召开国际会议,讨论软件危机问题,正式提出并使用"软件工程"的概念,标志着软件工程诞生。软件工程涉及与生产软件相关的所有活动,包括计算机科学、管理学、经济学、心理学等,其研究的主要内容是如何应用科学的理论和工程上的技术来指导软件的开发,从而达到以较少投资获得高质量软件的最终目标。

第四阶段是从20世纪80年代中期至今,客户端/服务器(Client/Server,C/S)的体系结构,特别是Web技术和网络分布式对象技术飞速发展,导致软件系统体系结构向更加灵活的多层分布式结构演变,CORBA、EJB、COM/DCOM等三大分布式的对象模型技术相继出现。2006年,面向服务架构(Service-Oriented Architecture,SOA)作为下一代软件架构,采用"抽象、松散耦合和粗粒度"的软件架构,根据需求,通过网络对松散耦合的粗粒度应用组件进行分布式部署、组合和使用,主要用于解决传统对象模型中无法解决的异构和耦合问题。

至此,软件发展经历了从Mainframe结构、C/S结构、浏览器/服务器(Browser/Server,B/S)多层分布式结构、SOA的演变过程,整个软件系统变得越来越分散、越来越开放、越来越强调互操作性。

1.1.2 软件项目

软件项目是一种特殊的项目,具有如下特点。

(1) 知识密集型,技术含量高。软件项目是知识密集型项目,技术性很强,需要大量高强度的脑力劳动。项目工作十分细致、复杂和容易出错。软件项目不需要使用大量物质资源,而主要使用人力资源,因此人员的因素极为重要,项目团队成员的结构、技能、责任心和团队精神对软件项目的成功与否有着决定性的影响。

(2) 涉及多个专业领域,多种技术综合应用。软件项目属于典型的跨学科合作项目,例如,开发大型管理信息系统就需要项目成员具有行业的业务知识、数据库技术、程序设计技术和信息安全技术等多专业领域知识。

(3) 项目范围和目标的灵活性。随着项目的进展,客户需求可能会发生变化,从而导致项目范围和目标的变化。软件开发不像其他产品的生产,有着非常具体的标准和检验方法,软件的标准柔性很大,衡量软件是否成功的重要标准就是用户满意度,但用户满意度这个标准在软件开发前很难精确地、完整地表达出来。

(4) 风险大,收益大。由于技术的高度复杂性和需求等因素的不确定性,软件项目风险控制难度较大,项目的成功率较低,但是一旦某个软件产品获得成功,将会带来相对高额的回报。

(5) 客户化程度高。项目的独特性在软件领域表现得更为突出,不同软件项目之间的差别较大。软件开发商往往要根据客户的具体要求提供独特的解决方案,即使有现成

的解决方案，也通常需要进行一定的客户化工作。

(6) 过程管理的重要性。软件项目需要对整个项目过程进行严格和科学的管理，尤其是对大型、复杂的软件项目。“质量产生于过程”，必须监控软件开发的过程和中间结果。没有严格的过程管理，开发人员的个人能力再强也没有用。

1.2 软件过程

软件过程是当前软件管理工程的核心问题。下面将介绍当前较流行的软件过程。其中，统一软件过程(Rational Unified Process，RUP)是一种当前企业应用较多的典型的软件过程模式，它是迭代增量、以架构为中心、用例驱动的软件开发方法。而敏捷过程作为轻量级开发过程，具有5个价值观：沟通、简单化、反馈、勇气、谦逊。

1.2.1 RUP

RUP是一种典型的软件过程模式，采用迭代增量式、以架构为中心和用例驱动的软件开发方法，以统一建模语言(Unified Modeling Language，UML)来描述软件开发过程。具体如下所示。

1. 软件生命周期的各个阶段

RUP中的软件生命周期在时间上被分解为4个顺序的阶段，分别是初始阶段、细化阶段、构建阶段和交付阶段。每个阶段结束于一个主要的里程碑，每个阶段本质上是两个里程碑之间的时间跨度。在每个阶段的结尾执行一次评估，以确定这个阶段的目标是否已经满足，如果评估结果满意，允许项目进入下一个阶段。其中每个阶段又可以进一步分解迭代。一个迭代是一个完整的开发循环，产生一个可执行的产品版本，作为最终产品的一个子集，产品增量式地发展，从一个迭代过程到另一个迭代过程，直到成为最终的系统。

如图1.1所示，RUP的过程用二维坐标来描述。横轴通过时间组织，是过程展开的生命周期特征，体现开发过程的动态结构，用来描述它的术语主要包括周期、阶段、迭代和里程碑；纵轴以内容来组织，是自然的逻辑活动，体现开发过程的静态结构，用来描述它的术语主要包括活动、产物、工作者和工作流。

2. RUP的核心工作流

RUP有9个核心工作流，分为6个核心过程工作流和3个核心支持工作流。尽管6个核心过程工作流可能使人想起传统瀑布模型中的几个阶段，但应注意迭代过程中的阶段是完全不同的，这些工作流在整个生命周期中一次又一次被执行。

1) 业务建模

业务建模工作流描述了如何为新的目标组织开发一个构想，并基于这个构想在业务用例模型和业务对象模型中定义组织的过程、角色和责任。

2) 需求

需求工作流的目标是描述系统应该做什么，并使开发人员和用户就这一描述达成共

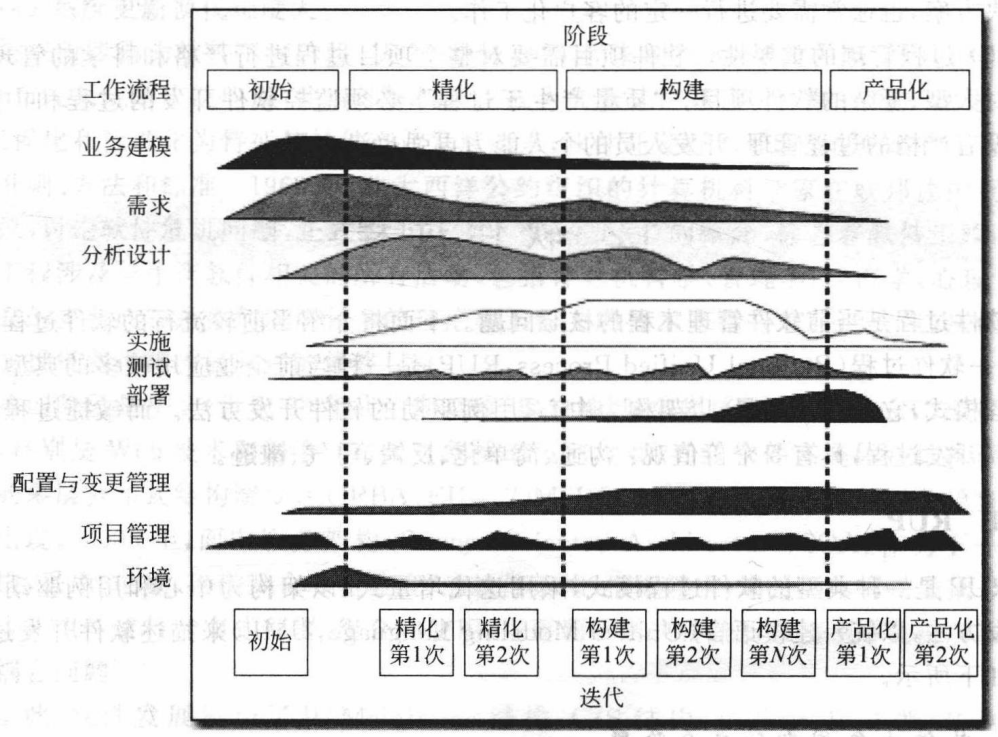


图 1.1 RUP 的过程图

识。为了达到该目标,要对需要的功能和约束进行提取、组织、文档化,最重要的是理解系统所解决问题的定义和范围。

3) 分析设计

分析和设计工作流是将需求转化成系统的设计,为系统开发一个健壮的结构,使其与实现环境相匹配。设计活动以体系结构设计为中心,其结果是一个设计模型和一个可选的分析模型。设计模型是源代码的抽象,由设计类和一些描述组成。设计类被组织成具有良好接口的设计包和设计子系统,而描述则体现了类的对象如何协同工作实现用例的功能。

4) 实施

实施工作流的目的包括以层次化的子系统形式定义代码的组织结构,以组件的形式(源文件、二进制文件、可执行文件)实现类和对象,将开发出的组件作为单元进行测试,集成单元使其成为可执行的系统。

5) 测试

测试从可靠性、功能性和系统性的三维模型来进行。测试工作流要验证对象间的交互作用,验证软件中所有组件的正确集成,检验所有的需求已被正确实现,识别并确认缺陷在软件部署之前被提出并处理。RUP 提出的迭代方法是在整个项目中进行测试,从而尽可能早地发现缺陷,从根本上降低修改缺陷的成本。

6) 部署

部署工作流的目的是成功地生成版本,并将软件分发给最终用户。部署工作流描述

了那些与确保软件产品对最终用户具有可用性相关的活动，它包括软件打包、生成软件本身以外的产品、安装软件、为用户提供帮助。在有些情况下，还可能包括计划和进行 beta 测试版、移植现有的软件和数据以及正式验收。

7）配置与变更管理

配置与变更管理工作流描绘了如何在多个成员组成的项目中控制大量的产物。配置和变更管理工作流提供了准则，以管理演化系统中的多个变体，跟踪软件创建过程中的版本。工作流描述了如何管理并行开发、分布式开发，如何自动化创建工程，同时也阐述了对产品的修改原因、时间、人员保持审计记录。

8）项目管理

软件项目管理平衡各种可能产生冲突的目标、管理风险，克服各种约束，并成功交付使用户满意的产品。其目标包括为项目的管理提供框架，为计划、人员配备、执行和监控项目提供实用的准则，为管理风险提供框架等。

9）环境

环境工作流的目的是向软件开发组织提供软件开发环境，包括过程和工具。环境工作流集中于配置项目过程中所需要的活动，同样也支持开发项目规范的活动，提供逐步的指导手册，并介绍如何在组织中实现过程。

3. *用例驱动是核心*

“用例驱动”是指开发过程遵循如下流程：它将按照一系列由用例驱动的工作流程来进行。首先是定义用例，然后是设计用例，最后，用例是测试人员构建测试用例的来源。用例在驱动整个软件开发过程，并且必须与系统体系结构协同开发。也就是说，用例驱动体系结构，而体系结构反过来又影响用例的选择。因此，随着生命期的继续，体系结构和用例都逐渐成熟。软件系统的体系结构也应成为软件过程的工作核心。

在每次迭代中，开发人员识别并详细定义相关用例，利用已选定的体系结构作为指导，来建立一个设计，以组件形式实现该设计，并验证这些组件满足了用例。如果一次迭代达到了它的目标，那么开发过程就进入下一次迭代的开发了。当一次迭代没有满足它的目标时，开发人员必须重新设计并加以实现。

1.2.2 敏捷过程

为了克服传统的软件生命周期模型，如瀑布模型、螺旋模型等在现代软件产业方面的局限性，2001 年，在研讨软件过程未来发展趋势的会议上，17 位业界专家就什么是“敏捷”达成一致意见，成立了“敏捷联盟”，并发布了《联盟敏捷宣言》（http://www.agilealliance.org/principles.html）。这份《联盟敏捷宣言》是“敏捷软件开发”价值和目标的浓缩定义，通过许多共同的原则进行了细化。敏捷开发过程的方法很多，主要有 Scrum，Crystal，特征驱动软件开发（Feature Driven Development，FDD），自适应软件开发（Adaptive Software Development，ASD）以及最重要的极限编程（eXtreme Programming，XP）。

极限编程是敏捷方法中最著名的，于 1998 年由 Smalltalk 社群中的大师级人物 Kent

Beck 首先倡导的,由一系列简单却互相依赖的实践组成。

敏捷过程定义了一系列核心原则和辅助原则,为软件开发项目中的建模实践奠定了基石。《联盟敏捷宣言》制定的原则如下。

(1) 我们最优先做的是通过尽早、持续地交付有价值的软件来使客户满意。

(2) 在项目的整个开发期间,业务人员和开发人员必须天天在一起工作。

(3) 即使到了开发后期,也欢迎需求变化。

(4) 经常性地交付可以工作的软件。

(5) 可以工作的软件是主要的进度度量标准。

(6) 围绕被激励起的个体来构建项目。为他们提供所需的环境和支持,并信任他们能胜任工作。

(7) 最好的架构、需求和设计来自自组织的团队。

(8) 在团队内部,最有效果和最有效率的传递信息的方法是面对面的交流。

(9) 敏捷过程提倡可持续的开发速度。

(10) 不断地关注最优秀的技术和良好的设计能增强敏捷能力。

(11) 简单是根本的。

(12) 每隔一定时间,开发团队都会对如何能有效地工作进行反省,然后相应地对自己的行为进行相应的调整。

1.3 软件测试

1.3.1 测试历程

软件测试伴随着软件的产生而产生。早在 20 世纪 50 年代,英国著名计算机科学家图灵就给出了软件测试的原始含义。他认为,测试是程序正确性证明的一种极端实验形式。早期软件开发过程中,软件规模小,复杂程度低,软件开发过程相当混乱无序,软件测试含义也比较窄,等同于“调试”,目的是纠正软件的故障,常常由软件开发人员自己进行。测试主要针对机器语言和汇编语言,设计特定的测试用例,运行被测试程序,将所得结果与预期结果进行比较,从而判断程序的正确性。对测试的投入极少,测试的介入也晚,常常是等到形成代码、产品已经基本完成时才进行测试。

直到 1957 年,软件测试首次作为发现软件缺陷的活动,与调试区分开来。1972 年,北卡罗来纳大学举行首届软件测试会议,John Good Enough 和 Susan Gerhart 在 IEEE 上发表《测试数据选择的原理》,确定软件测试是软件的一种研究方向。1975 年,John Good Enough 首次提出了软件测试理论,从而把软件测试这一实践性很强的学科提高到了理论的高度。1979 年,Glenford Myers 在《软件测试艺术》一书中提出“测试是为发现错误而执行的一个程序或者系统的过程”。

20 世纪 80 年代早期,软件和 IT 行业进入了大发展时期,软件趋向大型化、高复杂度,软件的质量越来越重要。一些软件测试的基础理论和实用技术开始形成,软件开发的方式也逐渐由混乱无序的开发过程过渡到结构化的开发过程,以结构化分析与设计、结构

化评审、结构化程序设计以及结构化测试为特征。软件测试的性质和内容也随之发生变化，测试不再是一个单纯的发现错误的过程，而是具有软件质量评价的内容。软件工程的概念逐步形成，软件开发模型产生。1983 年，Bill Hetzel 在《软件测试完全指南》中指出，测试是以评价一个程序或者系统属性为目标的任何一种活动，是对软件质量的度量。IEEE 给软件测试定义为“使用人工或自动手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果直接的差别。”这个定义明确指出，软件测试的目的是为了检验软件系统是否满足需求，软件测试不再是一次性的，也不只是开发后期的活动，而是与整个开发流程融合成一体。

20 世纪 90 年代，随着面向对象分析和面向对象设计技术的日渐成熟，面向对象软件测试技术逐渐受到人们重视。1989 年，Fiedler 从面向对象的测试与传统测试的不同点出发，提出了面向对象单元测试的解决方案，从而开始了面向对象软件测试的研究工作。1994 年 9 月，Communication OF ACM 出版了面向对象的软件测试专集，涉及了类测试、集成测试和面向对象软件的课测试性等问题。

1996 年，测试能力成熟度、测试支持度、测试成熟度等一系列软件测试相关理论提出。到了 2002 年，Rick 和 Stefan 在《系统的软件测试》一书中对软件测试作了进一步描述：测试是为了度量和提高软件的质量，对软件进行工程设计、实施和维护的整个生命周期过程。近 20 年来，随着计算机和软件技术的飞速发展，软件测试技术的研究也取得了很大突破。许多测试模型（如 V 模型等）产生，单元测试、自动化测试等方面涌现了大量的软件测试工具。在软件测试工具平台方面，商业化的软件测试工具，如捕获/回放工具、Web 测试工具、性能测试工具、测试管理工具、代码测试工具等产生很多，一些开放源码社区中也出现了许多软件测试工具，它们得到了广泛应用且相当成熟和完善。

1.3.2 测试与开发的关系

1. 瀑布模型与软件测试的关系

瀑布模型认为，测试是指在代码完成后、处于运行维护阶段之前，通过运行程序来发现程序代码或软件系统中错误。因此，如果需求和设计上存在缺陷问题，就会造成大量返工，增加软件开发的成本等。为了更早地发现问题，测试应延伸到需求评审、设计审查活动中，软件生命周期的每一阶段都应包含测试。瀑布模型与软件测试的关系如图 1.2 所示。

2. 螺旋模型与软件测试的关系

大型软件项目通常有很多不确定性和风险性，而且异常复杂，如果采用瀑布模型那种“一次性完成”的线性过程模型，项目失败的风险就很大，因此需要采用一种渐进式的演化过程模型——螺旋模型。螺旋模型将测试看做是前进的一步，并试图将产品分解成增量版本，每个增量版本都可以单独测试。螺旋模型与软件测试的关系如图 1.3 所示。

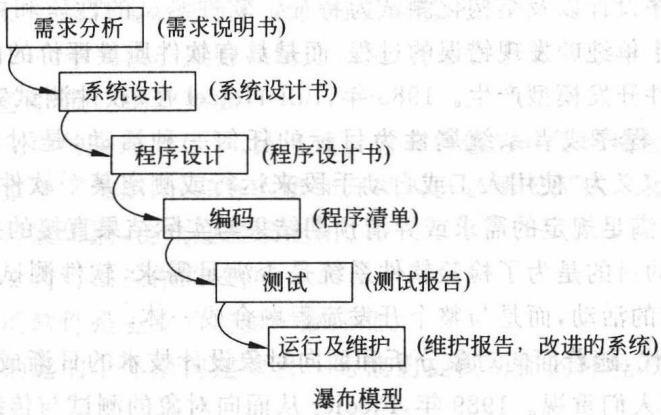


图 1.2　瀑布模型与软件测试的关系

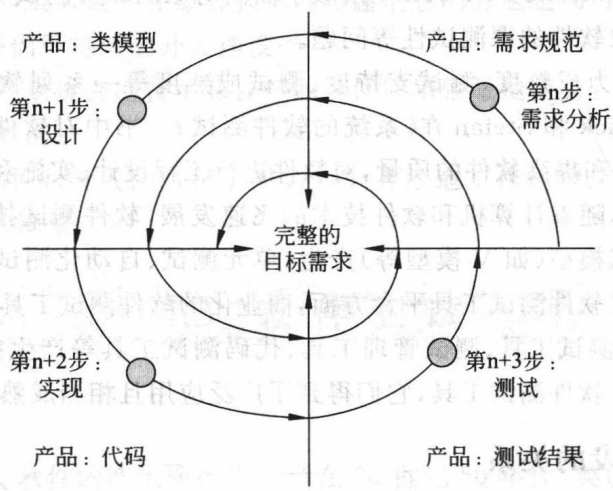


图 1.3　螺旋模型与软件测试的关系

1.4　软件缺陷

1.4.1　缺陷案例

1963 年，由于用 FORTRAN 程序设计语言编写的飞行控制软件中的循环语句 DO 5 I=1,3 误写为 DO 5 I=1.3，DO 语句少了一个逗号，结果导致美国首次金星探测飞行失败，造成价值 1000 多万美元的损失。

1979 年，新西兰航空公司的一架客机因计算机控制的自动飞行系统发生故障而撞在阿尔卑斯山上，机上 257 名乘客全部遇难。

1992 年 10 月 26 日，伦敦救护中心的计算机辅助发送系统刚启动就崩溃了，导致这个全世界最大的每天要接运 5000 多病人的救护机构全部瘫痪。

1994 年，美国迪士尼公司的《狮子王》软件在少数系统中能正常工作，但在大众使用的常见系统中不行。后来证实，迪士尼公司没有对市场上投入使用的各种个人计算机机