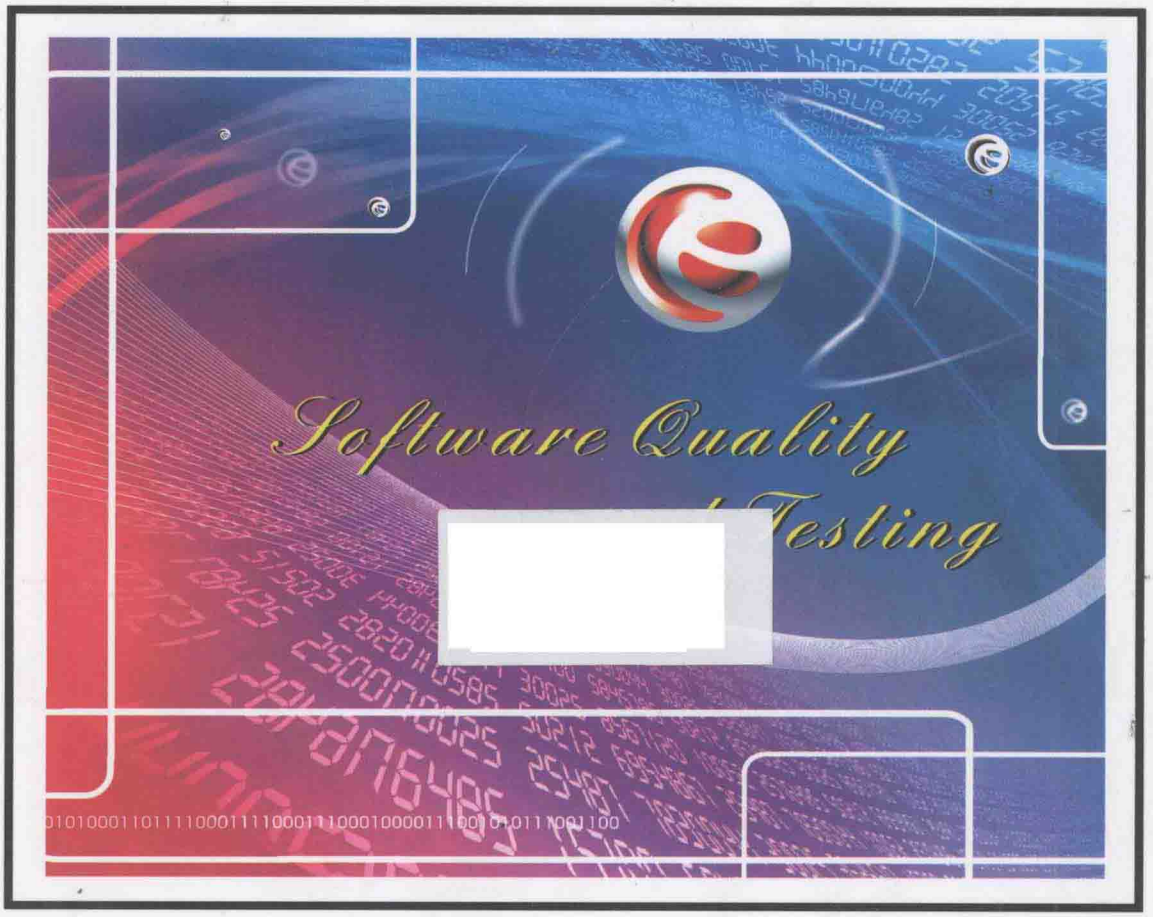
XSJ 高等学校计算机类"十二五"规划教材
COMPUTER

软件质量与测试

孟磊 主编
张姝 李航 孙阳 吴鹏 副主编



西安电子科技大学出版社
http://www.xduph.com

高等学校计算机类“十二五”规划教材

软件质量与测试

主　编　孟　磊
副主编　张　姝　李　航　孙　阳　吴　鹏

西安电子科技大学出版社

★ 内容简介 ★

本书共分 10 章。第 1 章阐述了软件测试发展史、软件测试的概念以及相关原则等方面的知识；第 2 章介绍了软件测试基本技术，包括白盒测试技术和黑盒测试技术；第 3 章至第 6 章按照软件测试流程分别详细介绍了单元测试、集成测试、系统测试和验收测试；第 7 章讨论了面向对象的软件测试；第 8 章介绍了 3 款主流的软件测试工具；第 9 章介绍了软件质量和质量保证；第 10 章介绍了软件测试管理的相关内容。

本书可作为高等院校计算机类专业的教材或教学参考书，也可作为计算机爱好者的自学用书。

图书在版编目(CIP)数据

软件质量与测试/孟磊主编. —西安：西安电子科技大学出版社，2015.3
高等学校计算机类“十二五”规划教材
ISBN 978-7-5606-3667-2

Ⅰ. ① 软… Ⅱ. ① 孟… Ⅲ. ① 软件质量—质量管理—高等学校—教材 ② 软件—测试—高等学校—教材 Ⅳ. ① TP311.5

中国版本图书馆 CIP 数据核字(2015)第 047907 号

策　　划　李惠萍
责任编辑　李惠萍　姚海军
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西华沐印刷科技有限责任公司
版　　次　2015 年 3 月第 1 版　2015 年 3 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印 张 16
字　　数　377 千字
印　　数　1～3000 册
定　　价　28.00 元
ISBN 978 - 7 - 5606 - 3667 - 2/TP
XDUP 3959001-1

前　言

当前，软件工程专业教学中普遍存在的问题是：学生在学习时，学的知识不知道干什么用；到应用时，用的知识不知道去哪里找；即使找到，也不知道具体应该怎么做。究其原因是现有教材大量地沿用了传统的计算机科学专业教材，在形式上更倾向于“学科性”导向的思路，侧重于理论知识的灌输，缺乏对学生技能的训练，从而偏离了软件工程专业“工程化”的核心。

解决这些问题的有效方法就是选用面向工作能力的知识与技能并重的教材，配合实训进行教学。

本书编写立足于“工程实践性”、“应用系统化”和“岗位胜任力”，在项目真实性、工程系统化、内容模块化和教辅支持度等方面都能很好地满足教学需求。

本书共分 10 章。第 1 章阐述了软件测试发展史、软件测试相关概念以及相关原则等方面的知识；第 2 章介绍了软件测试基本技术，包括白盒测试技术和黑盒测试技术；第 3 章至第 6 章按照软件测试流程分别详细介绍了单元测试、集成测试、系统测试和验收测试；第 7 章讨论了面向对象的软件测试；第 8 章介绍了 3 款主流的软件测试工具；第 9 章介绍了软件质量和质量保证；第 10 章介绍了软件测试管理的相关内容。

书中的每一章都是一个基本组织单元，每个单元都是面向工作能力的，以问题解决为中心。单元内容的组织结构包括：

- 学习目标，通过将学习目标分解为了解、理解、掌握三个认知层次，说明对知识的领会程度和对技能的掌握水平。学习目标的描述包括：了解事实和术语，理解概念，掌握原理、步骤和技术，具备应用策略、技巧和工具的技能。
- 知识点，包括陈述性知识和程序性知识两类。陈述性知识是显性知识，注重对理论性知识的储备，主要说明“是什么”(事实、术语)和“为什么”(概念、原理)；程序性知识是隐性知识，注重对实用性知识的掌握，主要说明“怎样做”(步骤、技术)和“怎样做得更好”(模式、经验)。
- 范例(个别章未给出)，通过对实际工作中案例的示范性说明，帮助学习者更好地领会知识点并获得解决问题的技能。示范性说明包括面对问题时知识应用的策略、技术运用的技巧和工具使用的方法。

• 习题，对本单元知识的掌握情况进行回顾测试，通过进一步练习提高学习者的技能应用水平。

本书由孟磊主编，张姝、李航、孙阳、吴鹏任副主编。本书的编写得到了沈阳师范大学软件学院全体同仁的大力支持，在此表示衷心感谢。

由于编者水平有限，时间仓促，尽管我们尽了最大的努力，但书中仍难免有不妥之处，恳请读者批评指正。

编　者

2014 年 11 月

目　录

第 1 章　软件测试概述

学习目标

- 了解软件测试的定义。
- 理解软件测试层次。
- 掌握软件测试研究的主要内容。

社会的不断进步和计算机科学技术的快速发展，使计算机软件在工业控制、医疗、通信、交通、金融、军事、航天航空等方面的应用越来越广泛和深入。作为计算机的主要组成部分，计算机软件在其中起着至关重要的作用，软件产品的质量是否符合要求自然成为人们共同关注的焦点。软件测试是保障软件质量的重要手段，也是软件从开发到使用的最后一道屏障，是软件工程的重要组成部分。运用软件测试提供的规范化的分析设计方法，可以尽量避免软件错误的发生和消除已经发生的 Bug，使程序中的 Bug 率达到尽可能低的程度，为最终消除软件危机提供强有力的技术保障。所以，随着软件工程技术的发展，软件规模增大，软件测试在软件开发过程中的作用显得越来越重要。

1.1　什么是软件测试

1.1.1　软件测试发展

软件测试是伴随着软件的产生而发展的。早期软件开发过程中，软件复杂程度较低，规模相对较小，软件开发过程混乱，相当随意。软件开发人员根本没办法区分“调试”和“测试”，往往把软件的调试过程就等同于测试了。另外，软件测试也不是贯穿于软件开发整个生命周期的，而是在代码已经编完了，象征性地测试一下，由于留给软件测试的时间已经非常少了，因此也起不了什么作用。

1972 年在北卡罗来纳大学举行了首届软件测试正式会议。1975 年 John Good Enough 和 Susan Gerhart 在 IEEE 上发表了《测试数据选择的原理》一文，将软件测试确定为一种研究方向。1979 年 Myers 在《The Art of Software Testing》中明确软件测试的目的是“find errors in software”，一个好的测试是“find errors that not been found”。20 世纪 80 年代早期，“质量”概念逐渐被人们关注，软件测试定义发生了很大改变，测试不单纯是一个发现错误的过程，而且包含软件质量保证的内容，为此也制定了若干标准。20 世纪 90 年代，测试工具的使用盛行起来。到了 2002 年，Rick 和 Stefan 在《系统的软件测试》一书中对软件测试做了进一步定义：测试是为了度量和提高被测软件的质量，对被测试软件进行的工程设计、实施和维护的整个生命周期过程。时至今日，软件测试已经成为软件开发必不可

少的一部分，软件测试部门也已经成为软件公司的常设部门。

1.1.2 软件测试定义

简单地说，软件测试就是利用测试技术和测试工具按照测试方案和流程对软件产品进行功能和性能测试，测试中可能要根据需要编写不同的测试工具，设计和维护测试系统，要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。

IEEE 对软件测试有如下定义："使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验其是否满足规定的需要或是弄清楚预期结果与实际结果之间的差别。"这个定义至少包含两层含义：其一就是测试一个软件系统除了人工手段外，有些情况还需要借助自动化的测试工具作为辅助；其二就是在进行软件测试之前要有非常明确的预期结果。

1.1.3 软件测试目标

软件测试是软件工程的重要组成部分，而测试的目的主要有两个方面：一是提高软件的质量；二是对软件进行验证和确认。

软件产品和其他产品一样具有产品的质量，质量好坏关系到产品能否达到最初设计的需求，满足用户使用和运行的需要。作为软件产品的最低质量要求，必须保证在设定的运行环境下，软件能正确运行和实现预定的功能。软件产品的质量有一定的特殊性，与硬件产品不一样，软件产品的质量是不能被直接测量的，只能通过对影响软件产品质量的各种因素的评估来间接体现软件产品的质量。

软件质量通常可以大体从以下三个方面进行衡量。

(1) 软件的功能：包括软件执行的正确性、可用性和一致性。软件实现的功能情况体现了软件对外界所展示出的外在质量并可以用软件需求规格说明书来衡量。

(2) 软件的性能：包括软件的灵活性、可重用性和可维护性，体现了软件产品是否有较好的可扩展性。因此，作为软件产品的测试，应尽可能涵盖这些相关的因素，来获得对软件产品质量的综合评价。当然，有不同用途的软件产品，对质量的要求也不尽相同。

(3) 软件的规范：包括执行效率、代码可测试性、文档记录的完备性和结构化程度，这些能标示出软件开发过程中质量控制的好坏。

对软件进行确认和验证是软件测试的另一个主要目标。IEEE 中对二者分别进行了定义。确认是指对系统或组件的初期评估过程，以确定软件产品在对应的开发阶段是否满足该阶段启动时所对应的要求。常见的确认流程有单元测试、集成测试、系统测试、验收测试等。验证则定义为对系统或组件的末期评估过程，以确定软件在开发过程中或结束阶段是否满足了特定的要求。常见的验证技术包括形式化方法、错误导入、依赖性分析、灾难分析和风险分析等方法。软件的确认和验证已融入到了整个开发过程中的各个阶段，对软件开发过程中的正确性保证有着重要的作用。

除了上述两个主要的测试目标外，软件的性能测试、可靠性测试以及安全测试也是经常被提及的测试内容。

软件性能测试通常涉及对资源的使用、吞吐量、反应时间、平均或最大等待时间等因

素。性能的测试可以帮助确定软件系统的瓶颈，以及对系统性能进行比较和评估。软件的可靠性则是指系统工作无故障的概率，是与软件许多方面相联系的。软件毕竟不同于硬件，对其可靠性进行直接的量化非常困难，而测试则是一种非常有效的取样方法，可用于对软件的可靠性进行测量和评估。测试可以获得失效数据，用于建立失效估计模型以进一步对数据进行分析，或通过当前可靠性的分析来对软件将来的行为进行预测。压力和负载测试是当前网络应用测试中常用的手段，用于测试在一定约束条件下系统所能承受的并发用户量，以确定系统的负载承受力。软件的质量、可靠性总是和软件的安全紧密相关的，软件中的缺陷可能被入侵者所利用形成安全漏洞。随着网络的发展，软件安全问题已越来越严重。安全测试的目标包括查找和消除软件中可能导致安全隐患的缺陷，验证安全措施的有效性，以及确定软件系统中易于遭受安全攻击的薄弱环节。

1.2　软件测试基础

1.2.1　软件测试主要内容

1．软件工程与软件测试

发达国家在发展软件的过程中曾经走过不少弯路，受过许多的挫折，至今仍然经受着“软件危机”的困扰。人们开发优质软件的能力大大落后于计算机硬件日新月异的进展和社会对计算机软件不断增长的需求，这种状况已经严重妨碍了计算机技术的进步。为了摆脱软件危机，一门新的学科——软件工程产生并逐渐发展起来。几十年来软件工程的发展大致经历了如下几个阶段。

第一阶段——软件危机。

20 世纪中期，计算机的使用刚刚从军用领域转向民用领域，那时编写程序的工作被视同为艺术家的创作。当时的计算机硬件非常昂贵，编程人员追求的是如何在有限的处理器能力和存储器空间的约束下，编写出执行速度快、代码少的程序。程序中充满了各种各样让人迷惑的技巧。这时的软件开发非常依赖于开发人员的聪明才智。

到了 20 世纪 60 年代，计算机的应用范围得到较大扩展，对软件系统的需求和软件自身的复杂度急剧上升，传统的软件开发方法无法适应用户在质量、性能等方面对软件的需求。这就产生了所谓的“软件危机”。

从 20 世纪 60 年代中期到 70 年代中期是计算机系统发展的第二个时期，这一时期软件开始作为一种产品被广泛使用，出现了“软件工厂”，专门应别人的需求写软件。这一阶段软件开发的方法基本上仍然沿用早期的个体化软件开发方式，但软件的数量急剧增多，软件需求日趋复杂，维护的难度越来越大，开发成本高得惊人，而失败的软件开发项目却屡见不鲜。“软件危机”就这样开始了！

“软件危机”使得人们开始对软件及其特性进行更深一步的研究，改变了早期对软件的不正确看法。早期那些被认为是优秀的程序常常很难被别人看懂，通篇充满了程序技巧。现在人们普遍认为优秀的程序除了功能正确、性能优良之外，还应该容易看懂、容易使用、

容易修改和扩充。

1968 年，北大西洋公约组织(NATO)的计算机科学家在联邦德国召开的国际学术会议上第一次提出了“软件危机”这个名词。概括来说，软件危机包含两方面问题：一是如何开发软件，以满足不断增长、日趋复杂的需求；二是如何维护数量不断膨胀的软件产品。

软件不同于硬件，它是计算机系统中的逻辑部件而不是物理部件。软件样品即是产品，开发过程也就是生产过程。软件不会因使用时间流逝而“老化”或“用坏”。软件具有可持续运行的行为特性，在编出程序代码并在计算机上试运行之前，软件开发过程的进度情况较难衡量，软件质量也较难评价，因此管理和控制软件开发过程十分困难。软件质量不能根据大量制造的相同物体的品质来度量，而是与每一个组成部分的不同物体的品质紧密相关，因此，在运行时所出现的软件错误几乎都是在程序开发时期就存在而一直未被发现的，改正这类错误通常意味着改正或修改原来的程序设计，这就在客观上使得软件维护远比硬件维护困难。软件是一种信息产品，具有可扩展性，不属于刚性生产，与通用性强的硬件相比，软件更具有多样化的特点，更加接近人们的应用。

随着计算机应用领域的扩大，99%的软件应用需求已不再是定义良好的数值计算问题，而是难以精确描述且富于变化的非数值型应用问题。因此，当人们的应用需求变化发展的时候，往往要求通过改变软件来使计算机系统满足新的需求，维护用户业务的延续性。为解决这个问题，1968 年 NATO 会议上首次提出“软件工程”(Software Engineering)的概念，提出把软件开发从“艺术”和“个体行为”向“工程”和“群体协同工作”转化。其基本思想是应用计算机科学理论和技术以及工程管理原则和方法，按照预算和进度，实现满足用户要求的软件产品的定义、开发、发布和维护的工程。自此，催生了一门新的学科——软件工程。

第二阶段——传统软件工程。

为迎接软件危机的挑战，人们进行了不懈的努力。这些努力大致上是沿着两个方向同时进行的。

一是从管理的角度，希望实现软件开发过程的工程化。这方面最为著名的成果就是提出了“瀑布式”生命周期模型(简称瀑布模型)。它是在 60 年代末“软件危机”后出现的第一个软件生命周期模型，如图 1-1 所示。

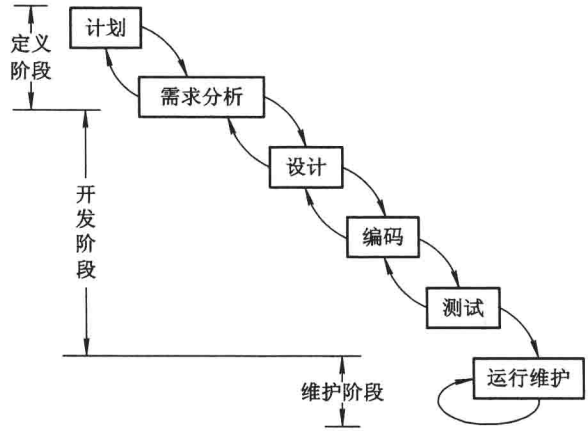


图 1-1　瀑布模型

后来，又有人针对该模型的不足，提出了快速原型法(见图 1-2)、螺旋模型(见图 1-3)、喷泉模型(见图 1-4)等，对“瀑布式”生命周期模型进行补充。现在，它们在软件开发的实践中被广泛采用。

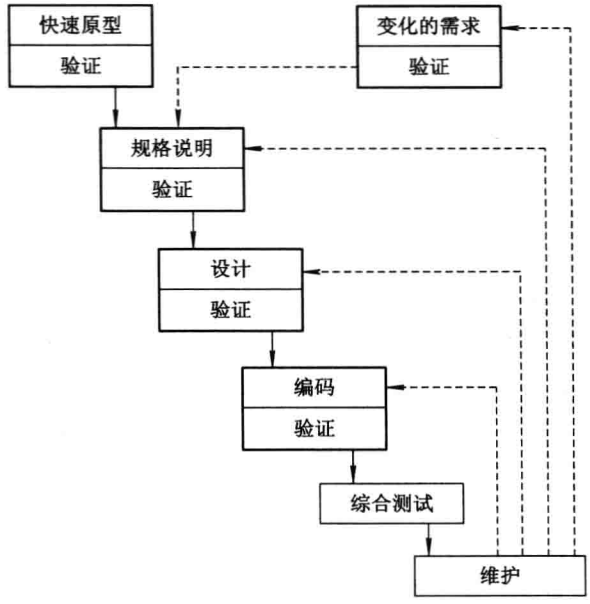


图 1-2　快速原型法

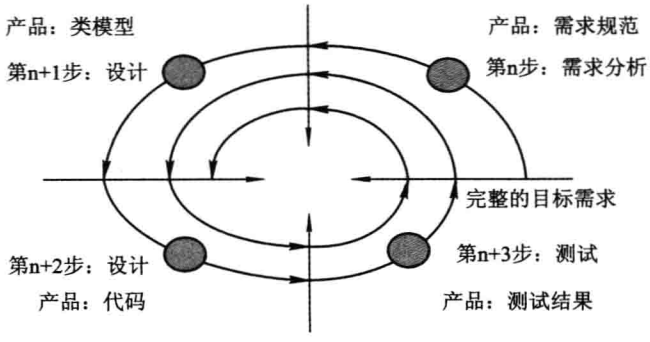


图 1-3　螺旋模型

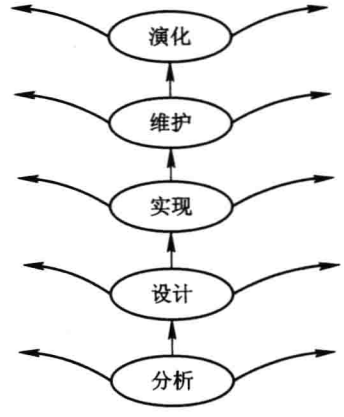


图 1-4　喷泉模型

软件工程发展的第二个方向，侧重于对软件开发过程中详细分析、设计方法的研究。这方面比较重要的成果就是在 20 世纪 70 年代风靡一时的结构化开发方法，即 PO(Procedure Oriented，面向过程的开发或结构化方法)以及结构化的分析、设计和相应的测试方法。

软件工程的目标是研制开发与生产出具有良好的软件质量和费用合理的产品。费用合理是指软件开发运行的整个开销能满足用户要求的程度，软件质量是指该软件能满足明确的和隐含的需求能力的有关特征和特性的总和。软件质量可用六个特性来作评价，即功能性、可靠性、易使用性、效率、可维护性、易移植性。

第三阶段——现代软件工程。

软件不是纯实物的产品，其中包含着人的因素，其中存在很多易变的因素，不可能像理想的物质生产过程，完全基于物理学等的原理来完成。早期的软件开发仅考虑人的因素，传统的软件工程强调物性的规律，而现代软件工程最根本的考虑就是人跟物的关系，就是人和机器(工具、自动化)在不同层次的不断循环发展的关系。

面向对象的分析、设计方法(OOA 和 OOD)的出现使传统的开发方法发生了翻天覆地的变化。随之而来的是面向对象建模语言(以 UML 为代表)、软件复用、基于组件的软件开发等新的方法和领域。

与上述开发方法相应的是从企业管理的角度提出的软件过程管理。即关注于软件生存周期中所实施的一系列活动并通过过程度量、过程评价和过程改进等对所建立的软件过程及其实例进行不断优化的活动，使得软件过程循环往复、螺旋上升式地发展。其中最著名的软件过程成熟度模型是美国卡内基梅隆大学软件工程研究所(SEI)建立的 CMM (Capability Maturity Model)，即能力成熟度模型。此模型在建立和发展之初，主要目的是为大型软件项目的招投标活动提供一种全面而客观的评审依据，而发展到后来，又同时被应用于许多软件机构内部的过程改进活动中。这在后面的章节中会详细讲解。

在软件开发过程中人们开始研制和使用软件工具，用以辅助进行软件项目管理与技术生产，人们还将软件生命周期各阶段使用的软件工具有机地集合成为一个整体，形成能够连续支持软件开发与维护全过程的集成化软件支援环境，希望从管理和技术两方面解决软件危机问题。

此外，人工智能与软件工程的结合成为 20 世纪 80 年代末期活跃的研究领域。基于程序变换、自动生成和可重用软件等软件新技术研究也已取得一定的进展，把程序设计自动化的进程向前推进一步。在软件工程理论的指导下，发达国家已经建立起较为完备的软件工业化生产体系，形成了强大的软件生产能力。软件标准化与可重用性得到了工业界的高度重视，在避免重用劳动、缓解软件危机方面起到了重要作用。

迄今为止，为了达到最初设定的目标，软件工程界已经提出了一系列的理论、方法、语言和工具，解决了软件开发过程中的若干问题，而软件工程正是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的一门新兴学科。

2．软件质量与度量

软件质量是一个模糊的、难以确定的概念。我们常常听说：这个软件好用，那个软件功能全、结构合理、层次分明、性能良好。这些模模糊糊的语言实在不能算作是软件质量评价，更加无法给软件质量一个科学的定量的评价。

在我国，软件产品质量低下，企业缺乏竞争力，更是一个长期困扰软件企业的问题。信息产业部在“十五”计划中，把提高软件产品质量、实现软件的产业化发展、加速软件产业和国际的接轨作为我国软件发展的重要目标。2000 年 6 月国务院颁发了《鼓励软件产业和集成电路产业发展的若干政策》。该文件的第五章第十七条明确提出：“鼓励软件出口型企业通过 GB/T19000，ISO9000 系列质量保证体系认证和 CMM(能力成熟度模型)认证。”

按照如上标准对软件进行评价主要涉及下面几点：

- 软件需求是衡量软件质量的基础，不符合需求的软件就不具备质量。设计的软件应

在功能、性能等方面都符合要求，并能可靠地运行。

- 软件结构良好，易读、易于理解，并易于修改、维护。
- 软件系统具有友好的用户界面，便于用户使用。
- 软件生存周期中各阶段文档齐全、规范，便于配置、管理。

随着竞争的日益激烈，国内软件企业越来越意识到提高软件产品质量和客户满意度对于企业生存和发展的战略意义，因此研究和应用软件质量度量技术，建立企业的软件质量保障体系已成为一个迫切而重要的任务。

1.2.2　软件测试过程模型

1. V 模型

传统的瀑布型软件开发中，仅仅把软件测试作为需求分析、概要设计、详细设计和编码之后的一个阶段，对如何进行软件测试没有进一步的描述。尽管有时测试工作会占用整个项目周期一半的时间，但仍然有人认为测试只是一个收尾工作，而不是主要的过程，V 模型就是针对瀑布型软件开发的一种补充。

V 模型是由 Paul Rook 于 20 世纪 80 年代后期提出的，并在英国国家计算中心文献中发布。V 模型在欧洲尤其是在德国得到了广泛的应用，也被德国定为信息技术系统的开发标准。V 模型可以说是瀑布模型的扩展，它根据瀑布模型的阶段来划分，对每个阶段进行有针对性的测试，由于这种划分方式简单易懂，因而 V 模型被广泛应用。值得注意的是，V 模型并不针对某种开发模型或是某种开发方法，它是按照软件生存周期中的不同阶段划分的，因此不能认为它只适合于瀑布模型。

早期的 V 模型展示了在软件生命周期中何时开始测试，准确地说，就是什么时候开始执行测试。它反映了测试活动与其他生命周期各活动的关系，描述了基本的开发过程和测试行为，非常明确地标明测试过程存在的不同级别，并清楚地描述了这些测试阶段和开发过程中各个阶段的对应关系。如图 1-5 所示，箭头代表了整个生命周期的流程，左边是开发过程各个阶段，右边是与之对应的测试过程各个阶段。

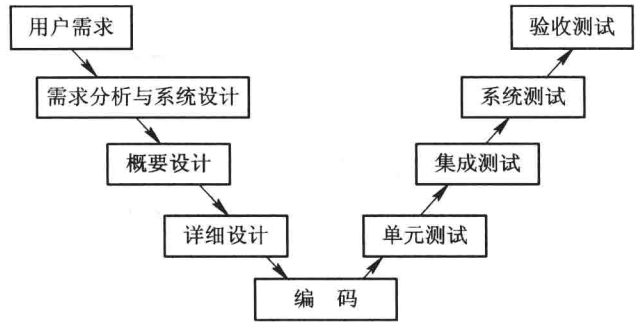


图 1-5　V 模型

由图 1-5 可知，软件测试不再是开发过程的一个收尾工作、一个可有可无的阶段了。测试工作要有组织分步骤以及系统地展开。所以 V 模型的贡献就是提高了软件测试地位，明确了各个阶段的内容。

2. W 模型

软件开发过程各个阶段都可能产生错误，根据国外学者统计，需求分析阶段产生的 Bug 占总 Bug 量的 60%以上，编码错误仅占 40%不到。另外，软件错误还具有传递性，即需求分析产生的 Bug 没有被及时发现，会依次传递到设计和开发中，并且被放大。比如，在分析设计时产生的错误，如果在编码结束后的测试中才被发现，其处理的代价约为在分析设计阶段发现和解决错误代价的 10 倍；如果这个错误在产品交付使用后才被发现和解决，其代价要超过 100 倍。因此，测试工作越早进行，发现和解决错误的代价越小，风险越小。根据这个观点，Evolutif 公司提出了相对于 V 模型更科学的 W 模型。

V 模型的局限性在于没有明确地说明对软件的早期测试，无法体现“尽早地和不断地进行软件测试”的原则。在 V 模型中增加软件各开发阶段应同步进行的测试，演化为 W 模型，如图 1-6 所示。在 W 模型中不难看出，开发是“V”，测试是与此并行的“V”。基于“尽早地和不断地进行软件测试”的原则，在软件的需求和设计阶段的测试活动应遵循 IEEE1012-1998《软件验证与确认(V&V)》的原则。W 模型是 V 模型的发展，强调的是测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、功能和设计同样要测试。测试与开发是同步进行的，从而有利于尽早地发现问题。

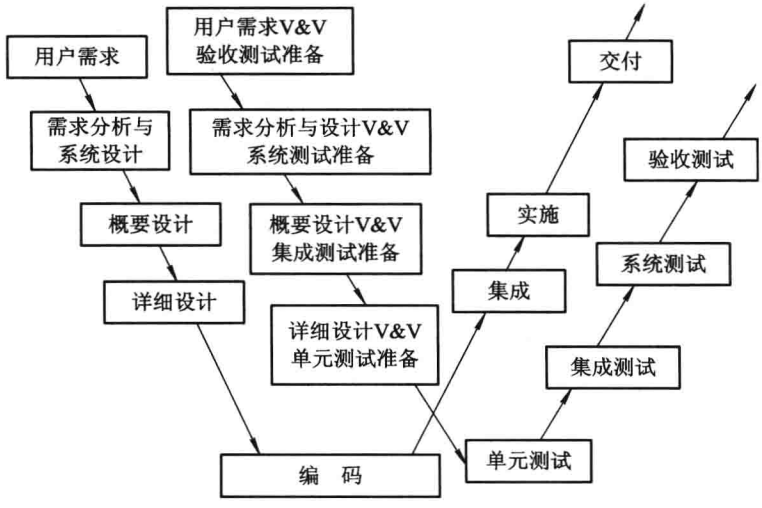


图 1-6　W 模型

W 模型是在 V 模型的基础上增加了需求测试、设计测试等，目的是确保需求的完整性、一致性、准确性和可测试性，以及设计对需求的可追踪性、正确性、规范性。也就是说软件需求、软件概要设计以及软件详细设计都需要进行测试。

根据 W 模型所述，一旦软件进入系统调研可行性分析阶段，就应该根据软件需求文档要求进行验收测试的设计，以便在交付时进行验收测试；在需求分析与系统设计阶段，就要进行需求分析测试，并进行系统测试设计，为接下来的系统测试做准备；在概要设计完成后，应对概要设计文档进行测试，并进行集成测试设计，以便进行集成测试；在进行详细设计后，应对详细设计文档进行测试，并进行单元测试设计，以便在模块完成后进行单元测试。

应用 W 模型就可以做到在每个阶段的开发活动完成后，即进行测试设计，测试和开发同步进行，将测试工作贯穿于软件开发的各个阶段，从而更有利于尽早发现问题。另外，通过 W 模型我们认识到软件测试不仅仅是程序测试，而应贯穿于软件开发的整个生命周期。在软件测试过程中，需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档都是软件测试的对象。

3. H 模型

不管是 V 模型还是 W 模型，都是把软件开发看作是需求、设计和编码等一系列的串行活动。而实际上，虽然这些活动之间存在互相牵制的关系，但在大部分时间内，它们可以交叉进行。虽然在软件开发过程中期望有清晰的需求、设计和编码阶段，但实践告诉我们，严格的阶段划分只是一种理想状态，相应的测试之间不存在严格的次序关系，同时各层次之间也存在反复触发、迭代和增量关系。

为了解决以上问题，有专家提出了如图 1-7 所示的 H 模型。H 模型中，软件测试过程活动完全独立，贯穿于产品的整个生命周期，与其他流程并发地进行。某个测试点准备就绪时，就可以从测试准备阶段进行到测试执行阶段。软件测试可以尽早地进行，并且可以根据被测对象的不同而分层次进行。

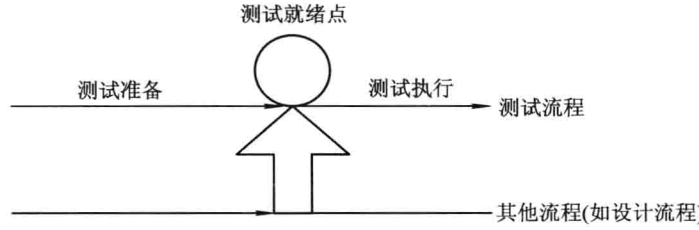


图 1-7　H 模型

图 1-7 中示意了在整个生产周期中某个层次上的一次测试“微循环”。图 1-7 中标注的其他流程可以是任意的开发流程，例如设计流程或者编码流程。也就是说，只要测试条件成熟了，测试准备活动完成了，测试执行活动就可以进行了。

H 模型揭示了一个原理：软件测试是一个独立的流程，贯穿产品整个生命周期，与其他流程并发地进行。H 模型指出软件测试要尽早准备，尽早执行。不同的测试活动可以是按照某个次序先后进行的，但也可能是反复的，只要某个测试达到测试就绪点，测试执行活动就可以开展。

采用 H 测试模型至少有以下三个好处：

(1) 有利于测试的分工，从而降低成本，提高效率。

首先，H 模型强调软件测试准备和测试执行分离。准备阶段和执行阶段有不同的测试活动。例如，测试准备活动包括测试需求分析、测试计划、测试分析、测试编码、测试验证等，而测试执行活动则包括测试运行、测试报告、测试分析等。准备阶段和执行阶段有不同的工作侧重点，不同的测试活动也需要不同的知识和技能。显而易见，测试的设计人员比执行人员有更高的能力要求。如果一个测试设计人员同时被指派去执行测试，那既是人力资源的浪费，也可能挫伤设计人员的创造性和积极性。所以，软件测试分工带来的第一个直接好处就是降低人力成本。第二个直接好处就是提高效率。分工带来的间接的长期