普通高等教育
软件工程
“十二五”规划教材

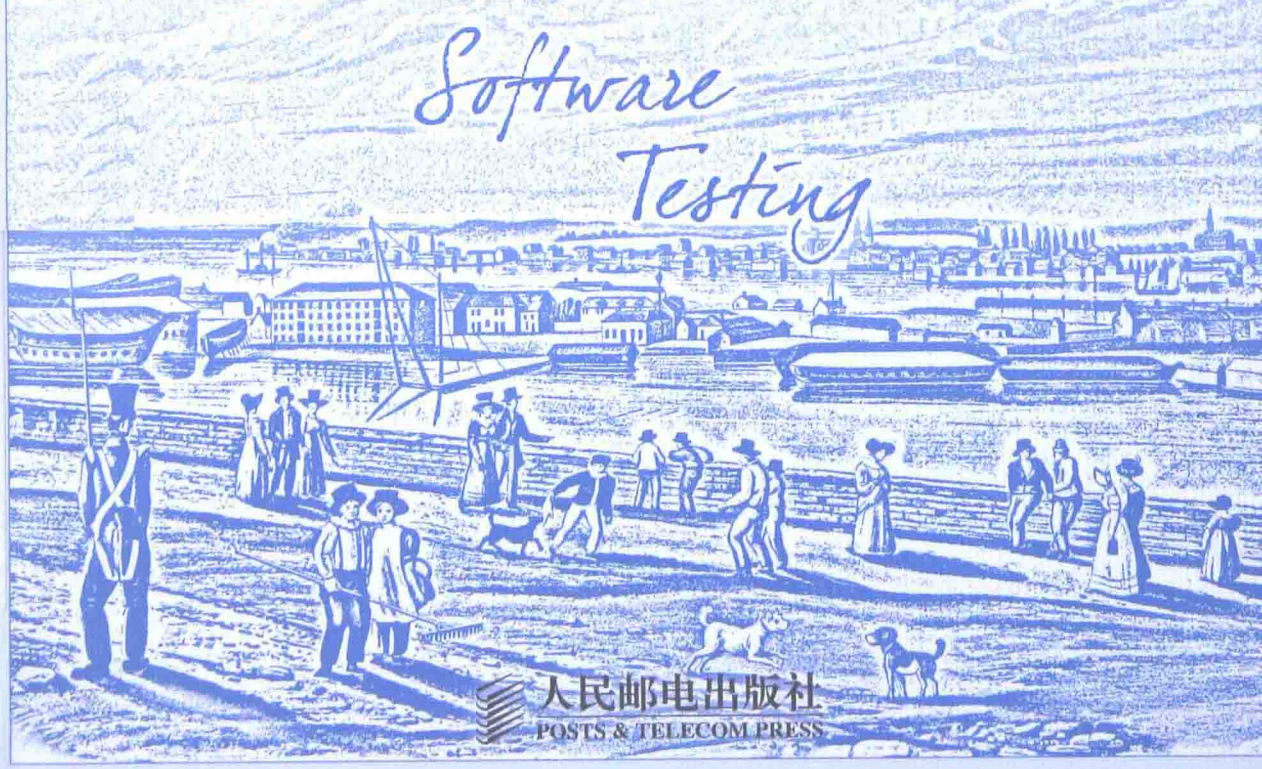
12th Five-Year Plan Textbooks
of Software Engineering

普通高等教育“十一五”
国家级规划教材

软件测试

（第2版）

佟伟光 ◎ 主 编
郭霏霏 ◎ 副主编



人民邮电出版社
POSTS & TELECOM PRESS

普通高等教育
软件工程 “十二五”规划教材

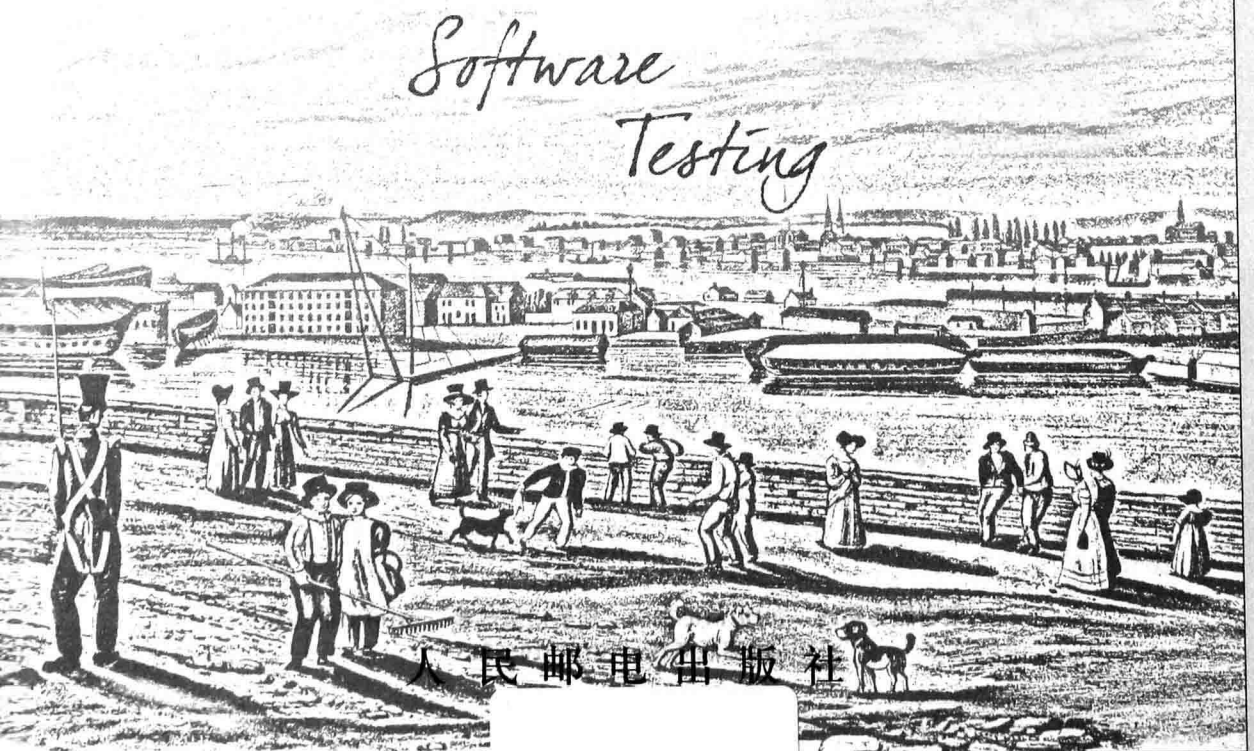
普通国家十一五

12th Five-Year Plan Textbooks
of Software Engineering

软件测试

（第2版）

佟伟光 ◎ 主　编
郭霏霏 ◎ 副主编



人民邮电出版社

图书在版编目（CIP）数据

软件测试 / 佟伟光主编. -- 2版. -- 北京 : 人民邮电出版社, 2015.1
普通高等教育软件工程“十二五”规划教材
ISBN 978-7-115-37465-3

Ⅰ. ①软… Ⅱ. ①佟… 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(2014)第273845号

内 容 提 要

本书第 1 版为普通高等教育“十一五”国家级规划教材，第 2 版教材对原书一些章节内容进行了重新编写，并增加了 Web 应用测试一章，将软件测试的新概念、新技术、新方法编入新教材中，使得教材内容更能体现软件测试技术的最新发展，更适合学生学习。本教材保持了教材原有内容的基本架构，特别注重突出教材的应用性、实践性，理论联系实际，把对学生应用能力的培养融汇于教材之中。第 2 版教材中保留了某大型软件公司一个完整的实际软件项目测试案例，并对该内容进一步充实，通过实例让学生了解在实际工作中如何实施软件测试，以此来实现巩固理论知识，提高学生实践能力的教学目标。

本书内容全面、注重实际、简明实用，例题、习题丰富，通俗易懂，易于学生学习，适合作为计算机、软件等相关专业软件测试课程教材，同时也可作为软件测试技术培训教材。

◆ 主　　编　佟伟光
副 主 编　郭霏霏
责任编辑　刘　博
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
大厂聚鑫印刷有限责任公司印刷

◆ 开本：787×1092　1/16
印张：17.5　　2015 年 1 月第 2 版
字数：472 千字　　2015 年 1 月河北第 1 次印刷

定价：39.80 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

目　录

第 1 章 软件测试概述

软件测试是软件工程中的重要部分，是确保软件质量的重要手段。最近几年，由于软件的复杂度不断增强、软件产业的不断发展，软件测试得到越来越广泛的重视。本章概括地介绍了软件测试的基本概念，包括软件测试的原则、分类和工作流程等基本知识。

1.1 软件、软件危机和软件工程

1.1.1 软件、软件危机和软件工程的基本概念

计算机系统分为硬件系统和软件系统两大部分。在过去的 50 多年里，随着微电子技术的发展和进步，计算机硬件技术以令人惊讶的速度发展，现在已经达到相当成熟的状态。

计算机软件是在计算机系统中与硬件相互依存的另一部分，它是包括程序、数据及其相关文档的完整集合。程序是指按特定的功能和性能要求而设计的能够执行的指令序列；数据是指程序能正常操纵、处理的信息及其数据结构；文档是指与程序设计开发、维护和使用有关的图文材料。

进入 20 世纪 60 年代，随着计算机技术的进步，软件功能日益复杂，人们对软件的需求急剧增加。软件开发从早期以个人活动为主的手工作坊方式，逐步转到以程序员小组为代表的集体开发方式。在这一转换过程中，国外的软件开发人员在开发一些大型软件系统时遇到了许多困难，有些系统最终彻底失败了；有些系统虽然完成了，但比原计划推迟了好几年，而且费用大大超过了预算；有些系统未能完全地满足用户的期望；有些系统则无法被修改和维护。例如，美国 IBM 公司的 OS/360 系统和美国空军某后勤系统都耗费了几千人·年的工作量，历尽艰辛，但结果却令人失望。落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象，这就是软件危机。软件危机主要表现在以下几个方面。

（1）软件生产不能满足日益增长的软件需求，软件生产率远低于硬件生产率和计算机应用的增长率，社会出现了软件供不应求的局面。更为严重的是，软件生产效率随软件生产规模的增加和软件复杂性的提高而急剧下降。

（2）软件生产率随软件规模与复杂性提高而下降，智力密集型行业的人力成本不断增加，这些都导致软件成本在计算机系统成本构成中的比例急剧上升。

（3）软件开发的进度与成本失控。人们很难估计软件开发的成本与进度，通常情况是预算成倍突破，项目计划一再延期。软件开发单位为了赶进度、节约成本，往往只有降低软件质量。软件开发陷入成本居高不下、质量无保证、用户不满意、开发单位信誉降低的怪圈中。

（4）软件系统实现的功能与实际需求不符。软件开发人员对用户需求缺乏深入的理解，往往急于编写程序，闭门造车，最后完成的软件与用户需求相距太远。

（5）软件难以维护。程序中的错误很难改正，要想使软件适应新的运行环境几乎不可能，软件在使用过程中不能增加用户需要的新功能，大量的软件开发人员在重复开发基本类似的软件。

（6）软件文档配置没有受到足够的重视。软件文档包括开发过程各阶段的说明书、数据词典、程序清单、软件使用手册、维护手册、软件测试报告和测试用例等。这些软件文档的不规范、不健全是造成软件开发的进度、成本不可控制和软件的维护、管理困难的重要原因。

软件危机实际上是软件开发与维护中存在的具有共性的种种问题的汇总。近40年来，为解决这些问题，计算机科学家和软件产业从业者已经做出了巨大的努力。

软件危机产生的原因可以从两个方面加以认识，一是软件产品的固有特性；二是软件专业人员自身的缺陷。

软件的不可预见性是软件产品的固有特点之一。与硬件产品不同，软件是计算机系统中的逻辑部件。在程序代码运行之前，开发工作的质量、进度难以度量。软件产品最终的使用价值是在软件运行过程中体现出来的。软件产品的故障隐蔽性强，可靠性难以度量，对原有故障的修改可能导致新的错误。

软件产品的固有特点之二是软件的规模较大并且逻辑较复杂。现代的软件产品往往规模庞大，功能多种多样、逻辑结构十分复杂。从软件开发管理的角度看，软件生产率常随软件规模和复杂性的增加而下降。就目前的软件技术水平而言，软件开发的工作量随软件规模的增大而呈几何级数上升。

来自于软件开发人员的弱点主要是，软件产品是人的思维结果，因此软件生产水平最终在相当程度上取决于软件人员的教育、训练和经验的积累；对于大型软件往往需要许多人合作开发，甚至要求软件开发人员深入应用领域的问题研究，这样就需要在用户与软件人员之间以及软件开发人员之间相互通信，在此过程中难免发生理解的差异，从而导致后续错误的设计或实现，而要消除这些误解和错误往往需要付出巨大的代价；另外，由于计算机技术和应用发展迅速，知识更新周期加快，软件开发人员经常处在变化之中，不仅需要适应硬件更新的变化，而且还要涉及日益扩大的应用领域问题研究，软件开发人员所进行的每一项软件开发几乎都必须调整自身的知识结构以适应新的问题求解的需要。

为了解决软件危机，既要有技术措施，又要有必要的组织管理措施。软件工程正是从技术和管理两个方面研究如何更好地开发和维护计算机软件的一门学科。

软件工程是应用计算机科学、数学及管理科学等原理开发软件的工程。通俗来说，软件工程是一套实现一个大型程序的原则方法，是将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，即按工程化的原则和方法组织软件开发工作。

1.1.2 软件工程的目标及其一般开发过程

从狭义上说，软件工程的目标是生产出满足预算、按期交付、用户满意的无缺陷的软件，进而当用户需求改变时，所生产的软件必须易于修改。从广义上说，软件工程的目标就是提高软件的质量与生产率，最终实现软件的工业化生产。

软件工程强调使用生存周期方法学，人类在解决复杂问题时，普遍采用的一个策略就是对问题进行分解，然后再分别解决各个子问题。软件工程采用的生存周期方法学就是从时间角度对软件开发和维护的复杂问题进行分解，把软件生存的漫长周期依次划分为若干个阶段，每个阶段有相对独立的任务，最后逐步完成每个阶段的任务。一个软件产品从形成概念开始，经过开发、测试、使用和维护，直到最后退出使用的全过程称为软件生存周期。软件工程采用的生存周期方法，对软件产品的质量保障以及组织好软件开发工具有重要意义。首先，由于能够把整个开发工作明确地划分成若干个开发步骤，这样就能把复杂的问题按阶段分别加以解决，使得对问题的认识与分析、解决的方案与采用的方法以及如何具体实现等工作在各个阶段都有着明确的目标。其次，

把软件开发划分成阶段，就为中间产品提供了检验的依据。一般软件生存周期包括问题的定义、软件开发、软件测试、软件使用与维护等几个阶段。

1. 问题的定义

问题的定义可分为软件系统的可行性研究和需求分析两个阶段，其基本任务是确定软件系统的工程需求。

可行性研究的任务是了解用户的要求及实现环境，从技术、经济和社会等几个方面研究并论证软件系统的可行性。参与软件系统开发的人员应在用户的配合下对用户的要求及系统的实现环境作详细的调查，并在调查研究的基础上撰写调查报告，然后根据调查报告及其他相关资料进行可行性论证。可行性论证一般包括技术可行性、操作可行性和经济可行性 3 个部分。在可行性论证的基础上制定初步的项目开发计划，大致确定软件系统开发所用的时间、资金和人力。

需求分析的任务是确定所要开发软件的功能需求、性能需求和运行环境约束，编制软件需求规格说明书、软件系统的确认测试准则。软件的功能需求应给出软件必须完成的功能，软件的性能需求包括软件的适应性、安全性、可靠性、可维护性、错误处理等，软件系统在运行环境方面的约束是指所开发的软件系统必须满足的运行环境方面的要求。软件需求不仅是软件开发的依据，也是软件验收的标准。因此，确定软件需求是软件开发的关键和难点，确定软件需求通常需要开发人员与用户多次反复沟通、讨论才能确认，完成需求分析工作是一项十分艰巨的任务。

2. 软件开发

软件开发是按照需求规格说明书的要求由抽象到具体、逐步完成软件开发的过程。软件开发一般由设计和实现等几个阶段组成，其中设计可分为概要设计和详细设计，主要是根据软件需求规格说明书建立软件系统的结构、算法、数据结构和各程序模块之间的接口信息，规定设计约束，为编写源代码提供必要的说明。实现也称为编码，就是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中，开发人员必须要制订统一、符合标准的编写规范，以保证程序的可读性和易维护性，提高程序的运行效率。

3. 软件测试

软件需经过严密的测试才能发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分单元测试、集成测试、系统测试以及验收测试 4 个阶段进行。测试的方法主要有白盒测试和黑盒测试。在测试过程中需要建立详细的测试计划并严格按照测试计划进行测试，以减少测试的随意性。大量统计表明，软件测试的工作量往往占软件开发总工作量的 40%以上，有时软件测试的成本甚至可高达软件工程其他步骤成本总和的 3 ~ 5 倍。

4. 软件使用和维护

软件的使用是指在软件通过测试后，将软件安装在用户确定的运行环境中移交给用户使用。软件的维护是指对软件系统进行修改或对软件需求变化作出反映的过程。当发现软件中的潜伏错误，或用户对软件需求进行修改，或软件运行环境发生变化时，都需要对软件进行维护。软件维护的成功与否直接影响软件的应用效果和软件的生存周期。软件的可维护性与软件的设计密切相关，因此在软件开发过程中应该重视对软件可维护性的支持。

软件生存周期的最后一个阶段是终止对软件系统的支持，软件停止使用。

1.1.3　软件过程模型

软件开发过程包含各种复杂的风险因素，为了解决由这些因素带来的种种问题，软件开发人员经过多年的摸索，总结出了多种软件工程的实现方式——软件过程模型，如瀑布过程模型、螺旋过程模型、增量过程模型、快速原型过程模、敏捷过程模型等。这些软件过程模型是软件开发的指导思想和全局性框架，它们的提出和发展反映了人们对软件过程的认识观，体现了人们对

软件过程认识的提高和飞跃。软件开发所遵循的软件过程是保证高质量软件开发的一个至关重要的因素。

1. 瀑布过程模型

瀑布过程模型反映了早期人们对软件工程的认识水平，是人们所熟悉的一种线性思维的体现。

瀑布过程模型强调阶段的划分及其顺序性、各阶段工作及其文档的完备性，是一种严格线性的、按阶段顺序的、逐步细化的开发模式，如图 1.1 所示。

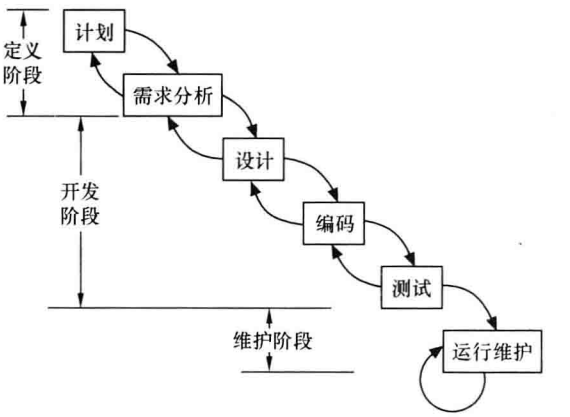


图 1.1　瀑布开发过程

瀑布过程模型主要由顺序的活动（或者阶段）组成，其开发阶段包括计划、需求分析、设计、编码、测试和运行维护等，各个阶段的主要工作内容如下。

- 计划阶段的工作主要是确定软件开发的总目标，给出软件的功能、性能、可靠性、接口等方面的设想；研究完成该项软件的可行性，探讨问题解决的方案；对可供开发使用的资源（如计算机硬、软件、人力等）、成本、可取得的效益和开发的进度作出估计；制定完成开发任务的实施计划。
- 需求分析是指收集产品的需求，对开发的软件进行详细的定义，由软件人员和用户共同讨论决定哪些需求是可以满足的，并且给予确切的描述；写出软件需求规格说明书等文档，提交管理机构审查。
- 设计是软件过程的技术核心。在设计阶段应把已确定的各项需求，转换成相应的体系结构，在结构中每一组成部分都是功能明确的模块，每个模块都能体现相应的需求，这一步骤称为总体设计。然后进行详细设计，对某个模块要完成的工作进行具体的描述，以便为程序编写打下基础。
- 编码阶段的工作是编写各种层次的代码，包括高级可视化开发系统生成的代码和用第四代程序设计语言编写的代码等。
- 软件测试主要是为了发现程序中的错误。
- 运行维护主要是对软件进行修复和改动，使它能够持续发挥作用。

在实际软件开发过程中，并不是严格按照图 1.1 中所示各阶段顺序执行的，因此过程中的各部分之间都有某种程度的重叠。造成这种重叠的原因是上述任何一个阶段都不可能在下一阶段开始之前完全结束。软件开发人员很少像图 1.1 所示的那样使用单纯的瀑布过程模型，除非是很小的项目或者是开发的产品与软件开发人员以前做过的项目类似，这主要是因为绝大多数软件都是复杂的、非线性的。

2. 螺旋过程模型

螺旋过程模型需要经历多次需求分析、设计、实现、测试这组顺序活动。这样做有多种原因，其中主要的原因是规避风险；另一个原因是在早期构造软件的局部版本时即交给客户以获得反馈；还有一个原因是为了避免像瀑布过程模型一样一次集成大量的代码。

螺旋过程模型的基本思路是依据前一个版本的结果构造新的版本，这个不断重复迭代的过程形成了一个螺旋上升的路径，如图 1.2 所示。

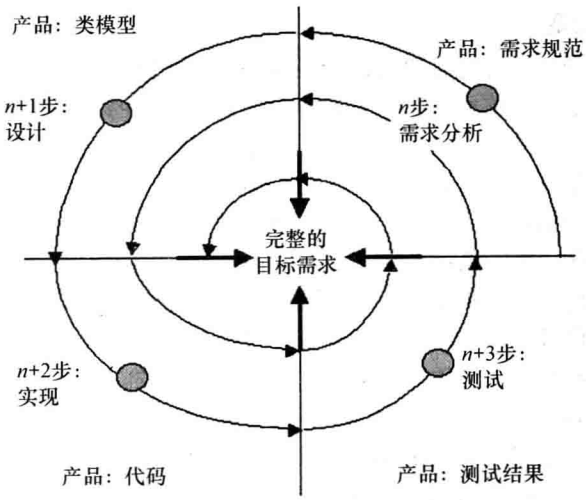


图 1.2　螺旋开发过程

螺旋过程模型的一个额外的优点就是能够在每次迭代中都收集到过程中产生的各种度量数据。例如，在第一次迭代时记录下小组进行设计和实现所耗费的时间，依此即可以改进后续设计和实现所耗费时间的估计方法，这对于没有任何历史数据的开发组织尤其 具有价值。

螺旋过程模型符合典型软件项目的发展特点，但是跟简单的瀑布过程模型相比，它需要投入更多的精力来更细致地管理其过程。这主要由于每次迭代完成之后都必须保证文档的一致性，特别是代码应该实现文档中描述的设计并且满足文档中记录的需求。此外，为了提高开发小组的生产效率，往往会在前一个迭代结束之前就开始一次新的迭代，这为协调文档的一致性增加了一定难度。

螺旋开发过程需要多少次迭代？这取决于具体的情况。例如，由 3 人组成的开发小组、耗时 4 个月的项目大概需要 2 次或者 3 次迭代，而项目若采用 5 次迭代，则所需要的管理费用通常会超过新增迭代所创造的价值。

3. 增量过程模型

当迭代的速度加快，每次迭代只是在前一次的基础上增加少量功能的时候，这种迭代过程就是增量开发过程。

增量过程模型是用一种几乎连续的过程小幅度地推进项目，如图 1.3 所示。增量过程模型在项目的后期尤其适用，比如当项目处于维护阶段，或者立项的产品与原先开发出来的产品结构极为相似。

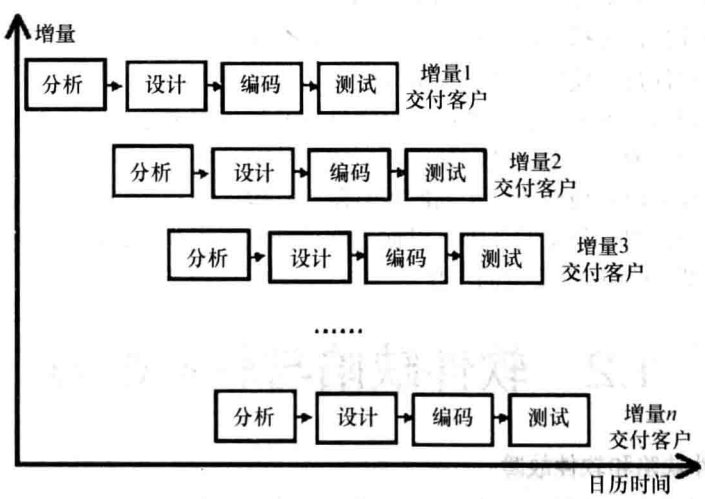


图 1.3　增量开发过程

4. 快速原型过程模型

在快速原型过程模型中，首先是快速进行系统分析，在设计人员和用户的紧密配合下，快速确定软件系统的基本要求，尽快实现一个可运行的、功能简单的原型系统，然后对原型系统逐步求精、不断扩充完善得到最终的软件系统。所谓“原型系统”就是应用系统的模型，用户在开发者的指导下试用原型，在试用的过程中考核评价原型的特性，分析其运行结果是否满足规格说明的要求，以及规格说明的描述是否满足用户的期望。开发人员根据用户的反馈意见纠正过去交互中的误解和分析中的错误，增补新的要求，并针对因环境变化或用户的新设想而引起系统需求的变动提出全面的修改意见。大多数原型系统中不合适的部分可以修正，修正后就成为新模型的基础，开发者和用户在迭代过程中不断将原型系统完善，直到软件的性能达到用户需求为止，因而快速原型过程模型能帮助开发人员快速完成所需要的目标系统。

快速原型过程模型的主要优点在于它是一种支持用户的方法，它使用户在系统生存周期的设计阶段起到积极的作用，并能减少系统开发的风险。特别是在大型项目的开发中，由于对项目的需求分析难以一次完成，应用快速原型过程模型效果更为明显。

5. 敏捷过程模型

是一种逐渐引起广泛关注的一些新型软件开发方法，由于其具有动态性且很容易适应环境。敏捷软件过程模型是一种迭代式增量软件开发过程。敏捷开发方法是一种以人为核心、迭代、循序渐进的开发方法。在敏捷开发中，软件项目的构建被切分成多个子项目，各个子项目的成果都经过测试，具备集成和可运行的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

敏捷软件开发是一个用来替代以文件驱动开发的瀑布开发模式。瀑布模型式是最典型的预见性的方法，严格遵循预先计划的需求、分析、设计、编码、测试的步骤顺序进行。步骤成果作为衡量进度的方法，例如需求规格，设计文档，测试计划和代码审阅等。瀑布式的主要的问题是它的严格分级导致的自由度降低，项目早期即作出的分析导致对后期需求的变化难以调整，代价高昂。瀑布式方法在需求不明并且在项目进行过程中可能变化的情况下基本是不可行的。

相对来讲，敏捷方法则在几周或者几个月的时间内完成相对较小的功能，强调的是能将尽早将尽量小的可用的功能交付使用，并在整个项目周期中持续改善和增强。

敏捷方法适用于小块工作，这些工作位于每次迭代以及迭代结尾发布的工作软件当中，敏捷方法的主要优势在于，它能完全适应用户环境，而且对产品进行持继迭代，它更注重交付能工作的软件，而不是实现需求规范中定义的需求。

以上5种模型只是众多软件过程模型中较为典型的，除此之外还有喷泉模型、统一软件开发过程模型等。介绍软件过程模型的目的是为了突出软件工程中软件过程模型的重要地位，从某种意义上说，不了解软件过程模型，就不了解软件工程。

同时还应该认识到，形成一套完整而成熟的软件开发过程不是一蹴而就的，它需要一个从无序到有序、从特殊到一般、从定性到定量、最后再从静态到动态的历程，或者说软件开发组织在形成成熟的软件过程之前必须经历一系列的探索阶段。因此，有必要建立一个软件过程成熟度模型来对过程作出一个客观、公正的评价，以促进软件开发组织改进软件开发过程，这就是所谓的软件能力成熟度模型（CMM）要做的工作。

1.2 软件缺陷与软件故障

1. 什么是软件缺陷和软件故障

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。软件开发是很复杂的工作，

开发人员很容易出现错误，虽然软件从业人员、专家和学者付出了很多努力，但软件错误仍然存在。因此大家也得到了一项共识：软件中存在错误是软件的一种属性，是无法改变的。所以通常说软件测试的目的就是为了发现尽可能多的软件缺陷，并期望通过改错来把缺陷消灭，最终提高软件的质量。

软件错误是指在软件生存期内的不希望出现或不可接受的人为错误，软件错误导致软件缺陷的产生。

软件缺陷是存在于软件（文档、数据、程序）之中的不希望出现或不可接受的偏差，软件缺陷导致软件在运行于某一特定条件时出现软件故障，这时软件缺陷被激活。

软件故障是指软件在运行过程中产生的不希望出现或不可接受的内部状态，对软件故障若无适当措施（容错）加以及时处理，就会使软件失效。

软件失效是指软件在运行时产生的不希望出现或不可接受的外部行为结果。

2. 软件缺陷和软件故障案例

（1）“千年虫”问题

在 20 世纪 70 年代，程序员为了节约内存资源和硬盘空间，在存储日期数据时，只保留年份的后 2 位，如“1980”被存储为“80”。但是，当 2000 年到来的时候，问题出现了。比如银行存款程序在计算利息时，应该用现在的日期“2000 年 1 月 1 日”减去当时存款的日期，比如“1979 年 1 月 1 日”，结果应该是 21 年，如果利息是 3%，每 100 元银行要付给顾客大约 86 元利息。如果程序没有纠正年份只存储 2 位的问题，其存款年数就变为 - 89 年，这样顾客反要付给银行 1288 元的巨额利息。所以，当 2000 年快要临近的时候，为了解决这样一个简单的设计缺陷，全世界付出了几十亿美元的代价。

（2）阿丽亚娜 5 型火箭发射失败

1996 年欧洲航天局阿丽亚娜 5 型火箭发射后 40S 火箭爆炸，发射基地 2 名法国士兵当初死亡，历时 9 年的航天计划严重受挫，震惊了国际宇航界。爆炸是由惯性导航系统软件技术和设计中的一个小失误引起的。

（3）“冲击波”病毒

2003 年 8 月 11 日，“冲击波”病毒首先在美国爆发，美国的政府机关、企业以及个人用户的成千上万的计算机受到攻击。随后，“冲击波”病毒很快在 Internet 上广泛传播，使十几万台邮件服务器瘫痪，给整个世界范围内的 Internet 通信带来惨重损失。

“冲击波”病毒仅仅是利用 Microsoft Messenger Service 中的一个缺陷，攻破计算机安全屏障，可使安装 Windows 操作系统的计算机崩溃。Messenger Service 的这个缺陷几乎影响当时微软公司所有的 Windows 系统，微软公司不得不紧急发布补丁包，修正这个缺陷。

（4）Windows 2000 中文输入法漏洞

Windows 2000 的交互式登录窗口存在“简体中文输入法状态识别”全漏洞，利用该漏洞，黑客可以使用交互式登录窗口从本地或远程得到一个 Local System 权限，利用该权限，黑客可以添加一个管理员账户，进而完全控制一台计算机，随后微软公司紧急发布补丁包，修正这个缺陷。

（5）金山词霸出现的错误

金山词霸 2003 和金山快译 2003 正式在全国各地上市以后，很多用户强烈批评这 2 款软件在某些词语翻译上的错误，并且当金山词霸 2003 的安装路径不按默认路径，或者用户使用其他以英文命名的目录路径进行安装时，系统就会出现安装完成以后无法取词和无法解释等多类错误，以至于金山公司在正式版发布几天之后就不得不发布补丁。

（6）北京奥运会门票系统故障

2007 年 10 月 30 日上午 9 点，北京奥运会门票面对境内公众的第二阶段预售正式启动。

由于瞬间访问数量过大造成网络堵塞，技术系统应对不畅，造成很多申购者无法及时提交申

请，为此，票务中心向广大公众表示歉意，并宣布暂停第二阶段的门票销售。

（7）2009年2月份Google的Gmail故障

2009年2月份Google的Gmail故障，Gmail用户几小时不能访问邮箱，应该算是最近因软件故障而受到广泛关注的事件。据Google后称，那次故障是因数据中心之间的负载均衡软件的Bug引发的。

（8）中国铁路网上订票出故障遭质疑

2012年1月为缓解数百万外出务工者春节回家买票难的问题，中国推出铁路网上订票系统，但该系统刚推出几分钟后就崩溃，这引起成千上万人的愤怒。

从上面几个典型的软件质量问题实例可以看出，由于软件本身特有的性质，只要软件存在一个很小的错误，就可能带来灾难性的后果。有错是软件的属性，而且是无法改变的，问题在于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。

3. 软件产生错误的原因

软件产生错误的原因很多，为了能够预防错误，需要了解错误产生的原因，具体地说，主要有如下几方面。

（1）软件的复杂性。软件是复杂的，因为它是思想的产物。随着计算机技术的进步，软件的功能、结构日益复杂，算法的难度不断增加，但是软件却要求高精确性，任何一个环节出了差错都会导致软件出现错误。正因为这个原因，软件缺陷总是会层出不穷。

（2）交流不够、交流上有误解或者根本不进行交流。软件是复杂的，当软件的规模达到一定程度的时候，个人已经无法实现，此时出现了团队，但是如何保证队员之间思想的一致性成了问题，人与人思想之间的差异是客观存在的，交流不够、交流上有误解或者根本不进行交流，这些都会导致软件在开发和维护过程中遇到一系列严重的问题。

（3）程序设计错误。像所有的人一样，程序员也会出错。有些错误可能是随机的，程序员通常是因为一时的疏忽而造成程序设计错误。

（4）需求变化。需求变化的影响是多方面的，客户可能不了解需求变化带来的影响，也可能知道但又不得不那么做。需求变化可能造成系统的重新设计，设计人员的日程需要重新安排，已经完成的工作可能要重做或者完全抛弃，而且需求变化可能会对其他项目产生影响，硬件需求也可能要发生变化。如果有许多小的改变或者一次大的变化，项目各部分之间已知或未知的依赖性可能会增强，进而导致更多问题的出现，需求改变带来的复杂性可能导致错误。

（5）时间压力。软件项目的日程表很难做到准确，很多时候需要预计和猜测。当最终期限迫近和关键时刻到来之际，错误也就跟着来了。

（6）代码文档贫乏。代码文档贫乏或者不规范的文档使得代码的维护和修改变得异常艰辛，其后果是带来许多错误。

（7）软件开发工具。可视化工具、类库、编译器、脚本工具等软件开发工具，都会将自身的错误带到应用软件中。

事实上，无论采用什么技术和什么方法，软件中仍然会有错。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，这些错误需要测试来发现，软件中的错误密度也需要测试来进行估计。

1.3 软件质量与质量模型

软件的质量是软件的生命，它直接影响软件的使用与维护。软件开发人员和用户都十分重视软件的质量，软件质量问题也是软件工程的核心问题之一。什么是软件质量？软件质量是一个复

杂的概念，不同的人从不同的角度来看软件质量问题，会有不同的理解。人们经常说某某软件好用，某某软件功能全、结构合理、层次分明、运行速度快等。这些模糊的语言实在不能算作是软件质量评价，特别不能算作是对软件质量科学的、定量的评价。其实，软件质量，乃至于任何产品质量，都是很复杂的事物性质。随着计算机软硬件技术的发展，人们对软件质量的理解不断深化，软件质量的标准也在不断改变。按照 ISO/IEC 9126-1991（GB/T 16260—1996）“信息技术软件产品评价质量特性及其使用指南”标准，对软件质量定义如下。

软件质量是与软件产品满足明确或隐含需求的能力有关的特征和特性的总和。其含义有以下 4 个方面。

- 能满足给定需求的特性。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好的质量。设计的软件应在功能、性能等方面都符合要求，并可靠地运行。
- 具有所期望的各种属性的组合的程度，即软件结构良好，合理地利用系统资源，易读、易于理解，并易于修改，方便软件的维护。
- 能满足用户综合期望的程度，软件系统具有友好的用户界面，便于用户使用。
- 软件的组合特性。软件生存周期中各阶段文档齐全、规范，便于配置管理。

软件质量是各种因素的复杂组合，软件质量因素也称为软件质量特性。质量特性反映了质量的本质，讨论一个软件的质量问题，最终要归结到定义软件的质量特性。

那么，影响软件质量的因素都有哪些？评价软件质量的标准是怎样的？换句话说，满足哪些标准才能保证软件产品具有好的质量？为了回答这些问题就需要一个易于理解的质量模型，来帮助评估软件的质量和对风险进行识别、管理。目前已有很多质量模型，它们分别定义了不同的软件质量属性，比较常见的 3 个质量模型是 McCall 模型、Boehm 模型和 ISO 9126，它们的共同特点是把软件质量特性定义成分层模型。比较普遍的质量特性模型是两层结构，第一层是按大类划分质量特性，叫做基本质量特性；第二层是每个大类所包含的子类质量特性；最后在各个类别的质量特性中一一列出对应的或相关的标准。

早在 1976 年，Boehm 等人就提出软件质量模型的分层方案。1979 年 McCall 等人改进了 Boehm 质量模型，又提出了一种软件质量模型，质量模型中的质量概念基于 11 个特性之上，这 11 个特性分别面向软件产品的运行、修正、转移，它们与特性的关系如图 1.4 所示。McCall 等人认为，特性是软件质量的反映，软件属性可用作评价准则，量化地度量软件属性可以判别软件质量的优劣。

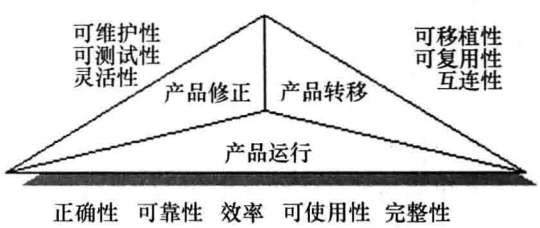


图 1.4　McCall 质量模型

对于没有开发经验的人来说，似乎自己开发的软件能够正确运行就可以了。但软件工程的结果是软件产品，一个产品的质量要求则是多方面的。实际的情况是，正确性只是反映软件质量的一个因素而已。软件的质量因素很多，如正确性、可靠性、可使用性、效率、完整性、可维护性、可测试性、灵活性、可移植性、可复用性、互连性等。

ISO 从更加普遍的用户角度出发，倡导并推动了关于统一软件质量特性认识的讨论，逐渐形成了比较一致的意见，并且不断改进。在 1991 年发布的 ISO/IEC 9126《软件质量特性与产品评价》中初步体现了这种国际性探讨的成果。ISO/IEC 9126—1991 标准规定的软件质量模型由 3 层组成，如图 1.5 所示。在这个模型中，第 1 层称为质量特性（SQRC），第 2 层称为质量子特性（SQDC），第 3 层称为度量（SQMC）。这个模型定义了 8 个质量特性，即正确性、可靠性、可维护性、效率、安全性、灵活性、可使用性、互连性；并推荐了 21 个子特性，如适合性、准确性等，但不作为标准。用于评价质量子特性的度量没有统一的标准，由各单位视实际情况制定，正如前面所提到的

那样，评价软件质量以用户需求为标准，一般都以用户的“满意度”进行衡量。

在1997年以后，ISO从软件生存周期角度考虑，提出了软件质量的度量概念。在这种关于衡量软件质量的概念的支持下，ISO修订了ISO/IEC 9126—1991，提出了一套新的9126系列标准，即ISO/IEC 9126-1、-2、-3和-4。2001年在新的ISO/IEC 9126《产品质量—质量模型》中定义的软件质量包括内部质量、外部质量和使用质量3部分，如图1.6所示。也就是说，“软件满足规定或潜在用户需求的能力”要从软件在内部、外部和使用中的表现来衡量。

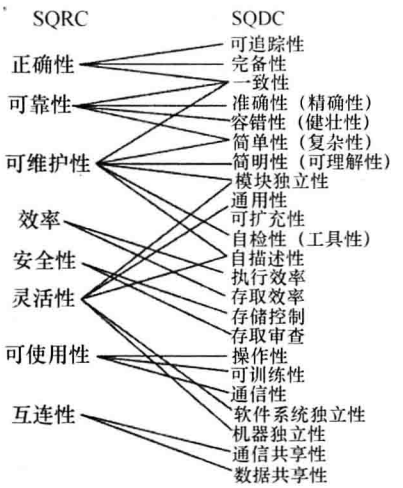


图1.5　ISO的软件质量评价模型

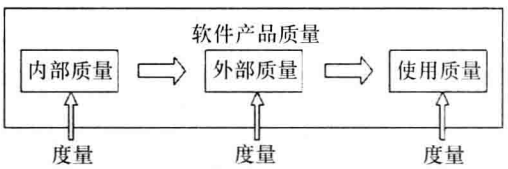


图1.6　新的ISO/IEC 9126软件质量模型

所谓的内部质量是从内部观点出发的软件产品特性的总体，是针对内部质量需求被测量和评价的质量。在新的ISO/IEC 9126《产品质量—质量模型》中，内部质量的定义是反映软件产品在规定条件下使用时满足需求的能力的特性，被视为在软件开发过程中（如在需求开发、软件设计、编写代码阶段）软件的质量特性。

内部质量特征主要包括以下几个方面。

- 可维护性，修改一个软件系统，提高其性能或修正其错误的能力。
- 灵活性，修改系统使其能适应于不同的用途或环境的能力，而不必对系统进行特定的设计。
- 可移植性，能修改所设计的某一部分，使其能在其他环境下运行的能力。
- 可重用性，能将系统的一部分用于其他系统的难易程度。
- 可读性，能读懂或理解系统源代码的能力。
- 可测试性，对整个系统进行单元或系统测试以证实其满足所有需求性能的测试难易程度。
- 可理解性，能从整个系统水平或细节上理解整个系统的难易程度。

外部质量的定义是软件产品在规定条件下使用时满足需求的程度。外部质量是从外部观点出发的软件产品特性的总体，它是当软件执行时，更典型的是使用外部度量在模拟环境中，用模拟数据测试时，所被测量和评价的质量，即在预定的系统环境中运行时可能达到的质量水平。

外部质量特征主要包括以下几个方面。

- 正确性，整个系统受说明、设计和实现的错误的影响程度。
- 可用性，用户学会和使用系统的难易程度。
- 效率，对系统资源的最小利用，包括存储时间和执行时间。
- 可靠性，系统在一定条件下执行特定功能的能力。
- 完整性，防止非法或不适当地访问的能力。
- 适应性，系统在应用或其他环境下不作修改就能使用的能力，而不必经过特定的设计。

- 精确性，系统不受错误影响的程度，尤其是在数据输出方面。精确性和正确性是不同的，精确性是对系统完成其工作情况的衡量，而不是它设计得是否正确。
- 坚固性，系统在无效输入或压力环境中能继续执行其功能的能力。

使用质量的定义是，在规定的使用环境下软件产品使特定用户在达到规定目标方面的能力。它是从用户观点出发，来看待软件产品用于特定环境和条件下的质量，反映的是从用户角度看到的软件产品在适当系统环境下满足其需求的程度。

使用质量用有效性、生产率、安全性、满意程度等质量特征表述。

对于一个实际的软件项目而言，想把上面的所有质量特征都做好是一件很难的事情。质量、资源和时间是项目管理的三要素，三者相互影响和制约，提高质量是有成本和时间代价的，提高质量可能带来更多资源的投入或进度的延后。因此，任何一个项目都应根据项目的实际特点来平衡好三要素，制订切实可行的质量目标。

任何形式的产品都是多个过程得到的结果，因此对过程进行管理与控制是提高产品质量的一个重要途径。对于一个软件项目，为了提高软件产品质量、缩短产品开发进度、节约产品开发成本，必须在软件开发过程的每个阶段、每个步骤上都要进行管理和控制。

1.4　软件测试

1.4.1　软件测试的概念

1. 软件测试的定义

什么是软件测试？简单地说，软件测试就是为了发现错误而执行程序的过程。软件测试是一个找错的过程，测试只能找出程序中的错误，而不能证明程序无错。软件测试要求以较少的用例、时间和人力找出软件中潜在的各种错误和缺陷，以确保软件的质量。

在 IEEE 所提出的软件工程标准术语中，软件测试被定义为："使用人工或自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或弄清楚预期结果与实际结果之间的差别。"软件测试是与软件质量密切联系在一起的，软件测试归根结底是为了保证软件质量。通常软件质量是以"满足需求"为基本衡量标准，IEEE 提出的软件测试定义明确提出了软件测试以检验是否满足需求为目标。

软件测试在软件生命周期中占据重要的地位，在传统的瀑布过程模型中，软件测试仅处于运行维护阶段之前，是软件产品交付用户使用之前保证软件质量的重要手段。近年来，软件工程界趋向于一种新的观点，即认为软件生命周期每一阶段中都应包含测试。由于软件工程采用的生存周期方法把软件开发划分成若干阶段，这样就对中间产品提供了检验的依据，各阶段完成的软件文档成为检验软件质量的主要对象。显然，表现在程序中的错误，并不一定是编码所引起的，很可能是详细设计、概要设计阶段，甚至是需求分析阶段的问题引起的。因此，即使针对源程序进行测试，所发现的问题的根源可能在开发前期的各个阶段，解决问题、纠正错误也必须追溯到前期的工作。正是如此，测试工作应该着眼于整个软件生命周期，特别是着眼于编码以前各个开发阶段的工作来保证软件的质量。也就是说，测试应该从软件生命周期的第一个阶段开始，并贯穿于整个的软件生命周期，从而检验各阶段的成果是否接近预期的目标，尽可能早地发现错误并加以修正。如果不在早期阶段进行测试，错误的延时扩散常常会导致最后成品测试的巨大困难。美国软件质量安全中心在 2000 年对美国 100 家知名的软件厂商进行统计，得出这样一个结论：软件缺陷在开发前期发现比在开发后期发现，在资金、人力上节约 90%；软件缺陷在推向市场前发现比在推出后发现，在资金、人力上节约 90%。所以说软件测试并非传统意义上产品交付前单一的