21世纪高等职业教育信息技术类规划教材
21 Shiji Gaodeng Zhiye Jiaoyu Xinxi Jishulei Guihua Jiaocai

软件测试技术
（第2版）

RUANJIAN CESHI JISHU

佟伟光 主编

- 由高校教师与测试工程师共同合作编写
- 全面介绍软件测试的基本知识和基本技术
- 提供实际软件项目的测试案例

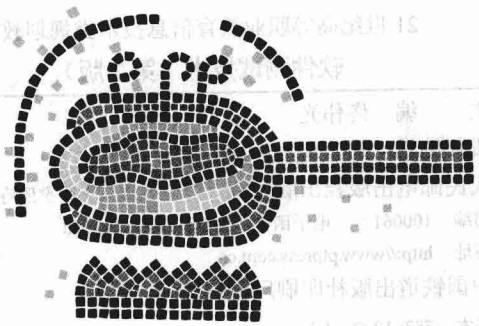
人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等职业教育信息技术类规划教材
21 Shiji Gaodeng Zhiye Jiaoyu Xinxi Jishulei Guihua Jiaocai

软件测试技术
（第2版）

RUANJIAN CESHI JISHU

佟伟光 主编



人民邮电出版社
北京

图书在版编目（CIP）数据

软件测试技术 / 佟伟光主编. -- 2版. -- 北京 : 人民邮电出版社, 2010.8
21世纪高等职业教育信息技术类规划教材
ISBN 978-7-115-22388-3

Ⅰ. ①软… Ⅱ. ①佟… Ⅲ. ①软件—测试—高等学校：技术学校—教材 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2010)第041304号

内 容 提 要

本书系统地介绍了软件测试的基本概念和基本知识，软件测试的基本技术、测试过程、测试用例设计、测试工具，如何报告软件缺陷，以及如何评估测试和测试项目管理等内容。本书内容由易到难，深入浅出，简明且通俗易懂，通过学习本书读者能够较好地掌握软件测试的基本知识和基本技术。另外，本教材的最后一章通过一个实际软件项目的测试案例，来加深读者对软件测试技术和软件测试过程的理解，尽最大努力使理论的应用更清晰、更形象。

本书适合作为高职高专院校软件测试课程的教材，以及软件测试培训班的教材，也可作为软件测试人员的自学参考书。

21世纪高等职业教育信息技术类规划教材

软件测试技术（第2版）

◆ 主　　编　佟伟光
责任编辑　刘　琦

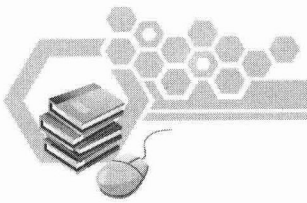
◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
中国铁道出版社印刷厂印刷

◆ 开本：787×1092　1/16
印张：16.75　　2010年8月第2版
字数：426千字　　2010年8月北京第1次印刷

ISBN 978-7-115-22388-3

定价：32.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154



前　言

本教材由高校教师、大型软件公司项目负责人及软件测试工程师共同合作编写。在教材编写过程中，融入了软件测试工程师的实践测试经验和教师授课经验。教材自 2005 年出版以来，受到广大读者的欢迎，也得到许多专家、教师和学生的热情支持和鼓励。几年来，我们又多次结合教学和测试实践，对软件测试课程教学进行了深入的研究，在此基础上修改并出版了本教材。本教材保持了第 1 版教材的基本架构，在继承通俗易懂、易于学习理解且实践性强等特点的基础上，对教材主要做了如下修订。

（1）对主要章节的内容进行了重新编写，充实、优化、调整了其余各章节的内容，将软件测试的新概念、新技术、新方法编入新教材中。在内容的安排上注意由易到难，深入浅出，简明且通俗易懂，使读者能够较好地掌握软件测试的基本知识和基本技术。

（2）将第 1 版教材的第 1、2 章合并为一章，并突出了软件测试基础知识的介绍，以使学生通过学习本课程，可以对软件测试的基本概念有比较全面的了解。

（3）由于软件测试基本技术和测试用例设计是测试的基础，本教材特别充实、优化了这两部分内容，并将这两部分内容分为第 2 章（软件测试基本技术）和第 3 章（测试用例设计）来编写，分别对白盒测试技术、黑盒测试技术和测试用例做了较详细的介绍，并给出了设计实例。

（4）精选和充实了教材每一章的习题，以方便学生复习，强化学生对重点内容的掌握，加深对所学内容的理解。

（5）本教材的编写特别注重突出应用性和实践性，理论联系实际。在测试实践一章保留了一个完整的实际软件项目测试案例，并对该内容做了适当的充实。实际测试案例的学习，将有助于强化学生的软件测试应用能力，实现巩固理论知识、提高实践能力的教学目标。

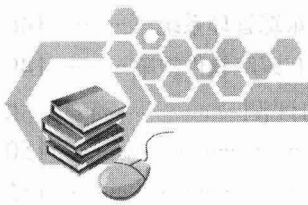
本书由佟伟光任主编、张欣任副主编，参加本书编写的还有宋喜莲、谢爽爽、赵忠诚、郑秀影、史江萍、杨胜等。

由于编者水平有限，书中难免出现不妥之处，请读者不吝指正。

编者的 E-mail 为 Weiguangt @ sina.com。

编　者

2009 年 12 月



目 录

第1章 软件测试概述

软件测试是软件工程的一个重要部分，是确保软件工程质量的重要手段。近几年来，由于软件工程的复杂度不断增强，随着软件的工业化发展趋势，软件测试受到了广泛的重视。本章概括地介绍了软件测试的基本概念、测试原则、分类和工作流程等基本知识。

1.1 软件开发过程

1.1.1 软件、软件危机、软件工程的基本概念

计算机系统分为计算机硬件和软件两大部分。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着，已经达到了相当成熟的状态。

计算机软件是计算机系统中与硬件相互依存的另一部分，它是程序、数据及其相关文档的完整集合。程序是按特定的功能和性能要求而设计的能够执行的指令序列，数据是程序能正常操作、处理的信息及其数据结构，文档是与程序设计开发、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，软件的规模和功能日益复杂，需求急剧增加。计算机软件开发从早期以个人活动为主的手工作坊方式逐步过渡到以程序员组形式为代表的集体开发方式。在这一过程中，国外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；有些系统未能完全满足用户当初的期望；有些系统则无法进行修改和维护。例如，美国IBM公司的OS/360系统和美国空军某后勤系统都花费了几十年的努力，历尽艰辛，但结果却令人失望。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，导致了软件生产与市场需求出现了极不适应的严重现象——软件危机。软件危机的主要表现如下。

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，出现了软件供不应求的局面。

② 软件生产率随软件规模与复杂性的提高而下降，智力密集造成人力成本增加，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 软件开发进度与成本失控。很难估计软件开发的成本与进度，通常是预算成倍突破，项目计划一再延期。软件开发单位为赶进度并节约成本，往往只有降低软件质量。软件开发陷入成本居高不下、软件质量无保证、用户不满、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，往往急于编程。闭门造车导致最后实现的系统与用户需求相去甚远。

⑤ 软件难以维护。程序中的错误很难改正，要使软件适应新的运行环境几乎不可能，软件使用过程中不能增加用户需要的新功能。而与此同时，大量的软件人员却在重复开发着基本类似的软件。

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、软件使用和维护手册、软件测试报告及测试用例等。这些软件文档的不规范、不健全是造成软件开发进程、成本不可控制，以及软件维护、管理、交流困难的重要原因。

软件危机的表现，实际上是软件开发与维护中存在的具有共性的问题。近30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。软件开发过程中，在程序代码运行之前，开发工作的质量和进度均难以度量。最终软件产品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，对原有故障的修改又可能导致新的错误。

软件产品的固有特点之二是软件的规模与逻辑复杂性。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。从软件开发管理角度看，软件生产率常随软件规模和复杂性的增大而下降。当多人合作完成一个系统时，作为一个工程项目，参与人员的组织与信息交流、工作质量与进度控制等更是一个复杂的问题。就目前的软件技术水平而言，软件开发工作量随软件规模呈几何级数上升。

软件开发人员的问题主要是没有掌握正确的软件开发方法，对软件的开发与维护存在许多模糊、错误的认识，不可避免地导致许多软件错误。软件管理人员的问题主要是软件管理技术落后，甚至缺乏软件质量管理。

宏观上，从整个社会对软件的需求来看，软件危机的实质是软件产品的供应赶不上需求的增长；微观上，“软件危机”简单地说就是开发的软件有错误，软件质量达不到要求，软件项目无法按时完成，软件项目的花费超预算。

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，软件工程是如何实现一个大型程序的一套原则方法，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。

为了很好地理解软件工程，可以将软件工程分解为人（people）、过程（process）、项目（project）和产品（product）4 个部分，即所谓的 4p 问题。4p 基本上给出了软件工程的一个大的轮廓。

- 人员是关于软件工程由谁来完成的问题。
- 过程指明了软件工程的实现方式，一般包括可行性与需求分析、系统设计、程序设计、测试和维护。
- 项目是指软件工程的实现过程，它包括和客户的交流、撰写文档、设计、编码和对产品做测试等一系列流程。
- 产品指所实现的应用软件及其相关的应用工件，应用工件包括需求分析所得到的软件需求说明书、详细设计的最终结果——设计模型、具体实现的源码和目标码以及测试过程和测试用例。

这 4p 之间的关系可表达为：每个项目的目标是产生一个软件产品，设计和实现一个有效产品是过程，而项目小组中人员的组织、协调、管理以及人与人之间的信息交流是影响项目成功与否的关键因素。

1.1.2　软件工程的目标及其一般开发过程

狭义来说，软件工程的目标在于生产出满足预算、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。广义来说，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，然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于软件产品的质量保障，以及组织好软件开发工具有着重要的意义。首先，把整个开发工作明确地划分成若干个开发步骤，就能够把复杂的问题按阶段分别加以解决，这使得问题的认识与分析、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，就对中间产品提供了检验的依据。一般软件生存周期包括软件定义、软件开发、软件测试、软件使用与维护等几个阶段。

1. 软件定义

软件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。

可行性分析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三个部分。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和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、可维护性和错误处理等。软件系统在运行环境方面的约束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，确定软件需求的过程通常需要开发人员与用户多次反复沟通、讨论，在需求分析阶段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的要求由抽象到具体，逐步生成软件的过程。软件开发一般由设计和实现等阶段组成。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建立软件系统的结构，明确算法、数据结构和各程序模块之间的接口信息，规定设计约束，并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。在程序编码中必须要制定统一且符合标准的编写规范，以保证程序的可读性和易维护性，以提高程序的运行效率。

3. 软件测试

软件必须经过严密的测试，以发现设计过程中存在的问题并加以纠正。把软件开发划分成阶段，为中间产品提供了检验的依据。因此，软件测试工作应该着眼于整个软件生命周期，测试应该从软件开发生命周期的第一个阶段开始，并贯穿于整个软件开发生命周期。

软件测试过程分为单元测试、集成测试、系统测试以及验收测试等。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，软件测试工作量往往占软件开发总工作量的40%以上，其成本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

软件的使用是在软件通过测试后，将软件安装在用户确定的运行环境中并移交给用户使用。软件的维护是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。软件的维护直接影响软件的应用和软件的生存期。软件的可维护性是与软件的设计密切相关的，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，即软件停止使用。

1.1.3 可供选择的软件过程模型

软件开发过程存在着各种复杂因素，为了解决由此带来的问题，软件开发者们经过多年的摸索，给出了多种实现软件工程的方式——软件过程模型，如瀑布过程模型、螺旋过程模型和增量过程模型等。这些软件过程模型是软件开发的指导思想和全局性框架，它的提出和发展反映了人

们对软件过程的某种认识观，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。

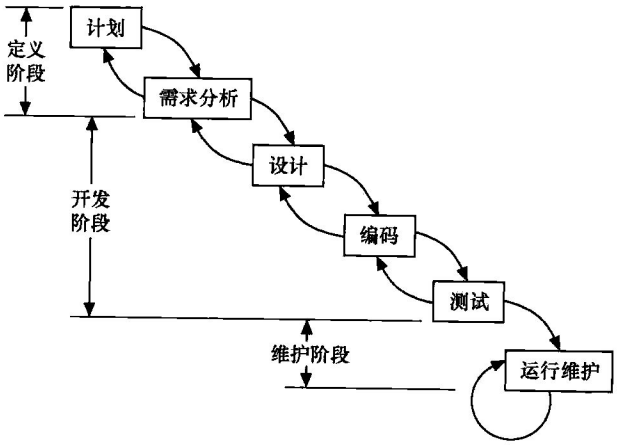


图 1.1　瀑布开发过程

该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。

- 计划阶段确定软件开发的总目标；给出软件的功能、性能、可靠性，以及接口等方面的设想；研究完成该项软件任务的可行性，探讨问题解决的方案；对可供开发使用的资源（如计算机硬件、软件以及人力等）、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。
- 需求分析阶段收集产品的需求；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；写出软件需求说明书等文档，提交管理机构审查。
- 设计阶段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，结构中的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。这一步称为总体设计。然后进行详细设计，对每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。
- 编码阶段的产品是各种层次的代码，包括由高级的可视化开发系统生成的代码和用第四代语言编写的代码等。
- 软件测试阶段用于发现程序中的错误。
- 运行维护阶段主要对应用进行修复和改动，使它能够持续发挥作用。

实际上，图 1.1 所示的各阶段并不是严格按照顺序执行的，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。

我们很少像图中所示的那样使用单纯的瀑布模型，除非是很小的项目或者是开发的产品与自己以前做过的项目类似。原因之一就是绝大多数应用系统都是复杂的、非线性的。

2. 螺旋过程模型

螺旋开发过程需要经历多次需求分析/设计/实现/测试这组顺序活动。这样做的主要原因是基于规避风险的需要，另一个原因是需要在早期构造产品的局部版本时即交给客户以获得反馈，最后是为了避免象瀑布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。

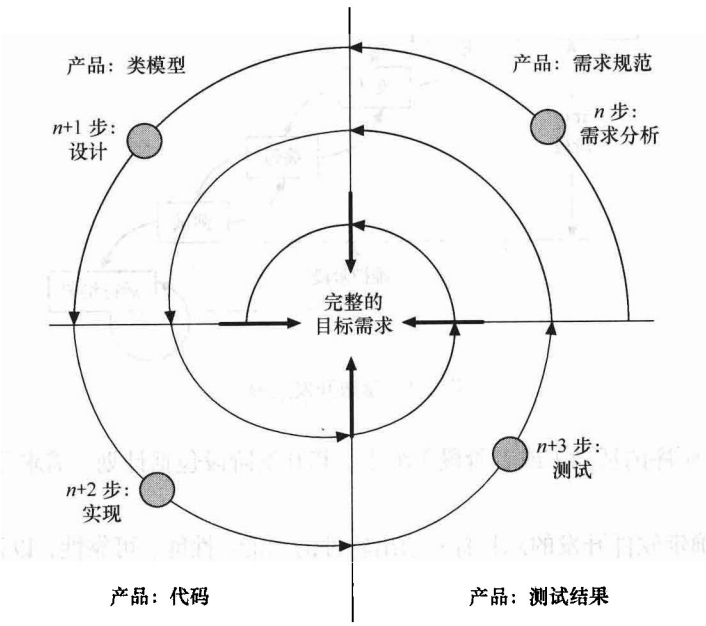


图 1.2　螺旋开发过程

迭代软件开发模型的另一个优点，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，依此即可改进后续设计和实现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有用。

螺旋过程符合典型项目的发展特点，但是跟简单的瀑布过程相比，它需要投入更多的精力来更细致地管理其过程。一个最直接的原因是，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。此外为了提高小组的生产效率，往往会在前一次迭代结束之前就开始新一次的迭代。这也为协调文档的一致性增加了难度。

螺旋开发过程需要多少次迭代？这取决于具体的情况。一个典型的工作量为 3 人/月，耗时 4 个月的项目大概需要两次或者三次迭代。项目若采用 5 次迭代，则所需要的管理费用通常会耗掉新增迭代所创造的价值。

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，我们就把这种迭代过程称为增量开发过程，如图 1.3 所示。

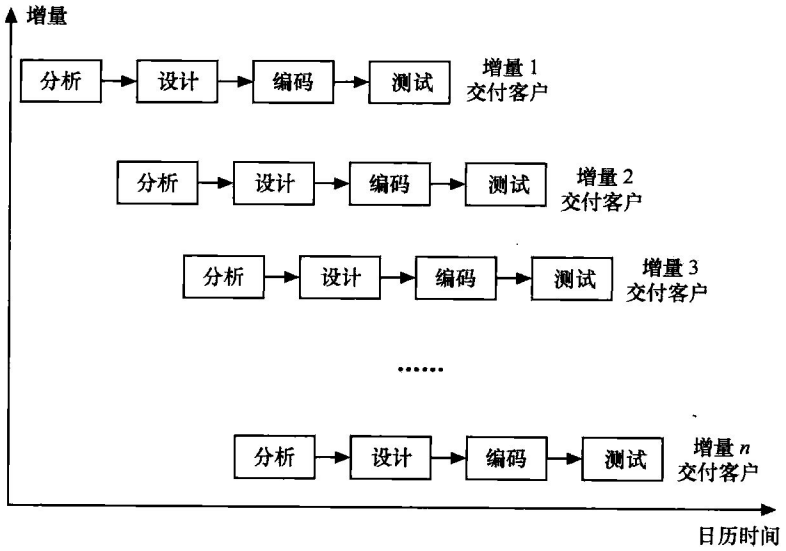


图 1.3　增量开发过程

3. 增量过程模型

有些时候可能会用一种几乎连续的过程小幅度地推进项目。这种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的时候。

4. 快速原型过程模型

快速原型过程模型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，然后通过对原型系统逐步求精，不断扩充完善得到最终的软件系统。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愿望，纠正过去交互中的误解和分析中的错误，增补新的要求，并为满足因环境变化或用户的新设想而引起系统需求的变动而提出全面的修改意见。大多数原型不合适的部分可以修正，以成为新模型的基础，开发者和用户在一次次的迭代过程中不断将原型完善，直到它的性能达到用户需求为止。由此可见，这个工作模型很快就能转换成所需的目标系统。

快速原型模型的主要优点在于它是一种支持用户的方法，使得用户在系统生存周期的设计阶段起到积极的作用。它能减少系统开发的风险，特别是在大型项目的开发中，由于对项目需求的分析难以一次完成，应用原型法效果更为明显。

以上四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。这里介绍软件过程模型只是为了突出软件工程中软件过程模型的重要地位，不了解软件过程模型，就不了解软件工程。

同时，我们还要认识到，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，或者说软件机构在形成成熟的软件过程之前必须经历一系列的成熟阶段。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，即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。软件开发是个很复杂的过程，期间很容易产生错误，无论软件从业人员、专家和学者做了多大的努力，软件错误仍然会存在。因而大家也得到了一种共识：软件中残存着错误，这是软件的一种属性，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缺陷，并期望通过改错来把缺陷消灭，以期提高软件的质量。

软件错误是指在软件生存期内的不希望或不可接受的人为错误，其结果是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）之中的那些不希望或不可接受的偏差。其结果是软件运行于某一特定条件时会出现软件故障，这时称软件缺陷被激活。

软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态，此时若无适当措施（容错）加以及时处理，便产生软件失效。

软件失效是指软件运行时产生的一种不希望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）美国迪斯尼公司的狮子王游戏软件兼容性问题

1994 年圣诞节前夕，迪斯尼公司发布了第一款面向儿童的多媒体光盘游戏“狮子王童话”，由于迪斯尼公司的著名品牌和事先的大力宣传及良好的促销活动，市场销售情况非常不错，该游戏成为父母为自己孩子过圣诞节的必买礼物。

但由于迪斯尼公司没有对当时市场上的各种 PC 机型进行完整的系统兼容性测试，只是在几种 PC 机型上进行了相关测试，导致这个游戏软件只能在少数系统中正常运行，但在大众使用的其他常见系统中却不能正常安装和运行。

（2）美国航天局火星登陆事故

1999 年 12 月 3 日，美国航天局发射的火星登陆飞船在试图登陆火星表面时失踪。

从理论上看，登陆计划是这样的：在飞船降落到火星的过程中，降落伞将被打开，减缓飞船的下落速度；降落伞打开后的几秒钟内，飞船的 3 条腿将迅速撑开，并在预定地点着陆；当飞船离地面 1 800m 时，它将丢弃降落伞，点燃登陆推进器，在余下的高度缓缓降落地面。

美国宇航局为了省钱，简化了确定何时关闭推进器的装置。为了替代其他太空船上使用的贵重雷达，在飞船的脚上装了一个廉价的触点开关，在计算机中设置一个数据位来关掉燃料。很简单，飞船的脚不“着地”，引擎就会点火。不幸的是，质量管理小组在事后的测试中发现，当飞船

的脚迅速撑开准备着陆时，机械震动在大多数情况下也会触发着地开关，从而设置错误的数据位。设想飞船开始着陆时，计算机极有可能关闭推进器，而火星登陆飞船下坠 1 800m 之后冲向地面，必然会撞成碎片。

为什么会出现这样的结果？原因很简单，登陆飞船经过了多个小组测试，其中一个小组测试飞船的脚落地过程，但从没有检查那个关键的数据位，因为那不是这个小组负责的范围；另一个小组测试着陆过程的其他部分，但这个小组总是在开始测试之前重置计算机、清除数据位。双方本身的工作都没什么问题，就是没有合在一起测试，其接口没有被测，而问题就在这里，后一个小组没有注意到数据位已经被错误设定。

仅仅由于两个测试小组单独进行测试，没有进行良好沟通，缺少一个集成测试的阶段，结果导致 1999 年美国宇航局的火星基地登陆飞船在试图登陆火星表面时突然坠毁失踪。

（3）跨世纪“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，在存储日期时，只保留年份的后两位，如“1980”被存为“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”减去最初存款的日期，比如“1989 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两位的问题，其存款年数就变为–89 年，变成顾客反要付给银行 1288 元的巨额利息。所以，当 2000 年快要来到的时候，为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。

（4）“冲击波”计算机病毒

2003 年 8 月 11 日，“冲击波”计算机病毒首先在美国发作，使美国的政府机关、企业及个人用户的成千上万的计算机受到攻击。随后，冲击波病毒很快在因特网上广泛传播，结果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”计算机病毒仅仅是利用了微软 Messenger Service 中的一个缺陷，攻破计算机安全屏障，使基于 Windows 操作系统的计算机崩溃。该缺陷几乎影响了当前所有的微软 Windows 系统，随后微软公司不得不紧急发布补丁包，以修正这个缺陷。

（5）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”安全漏洞，利用该漏洞，黑客可以在交互式登录窗口从本地或远程得到一个 Local System 权限，该权限使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，以修正这个缺陷。

（6）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来，很多用户强烈批评金山在某些词语翻译上的错误，以及当安装路径不是默认路径或者以其他英文路径进行安装时，就会出现安装完成以后无法取词和无法解释的现象等，以至于金山在正式版发布几天后就不得不发布补丁。

从上面几个典型的软件质量问题实例可以看出，软件本身特有的性质决定了只要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

给软件带来错误的原因很多，为了能够预防错误，必须了解错误产生的原因。具体地说，主要有如下几点。

（1）软件复杂性

软件是复杂的，因为它是思想的产物。计算机技术的进步，使软件规模、功能、结构日益复杂，算法的难度不断增加，而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，当软件的规模足够大时，个人已经无法实现，此时便出现了团队，但是如何保证队员之间思想的一致性就成了问题。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，会导致软件的开发和维护过程中出现一系列严重问题。

（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，它们通常是因为一时的疏忽造成的。

（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。需求变化的后果可能会造成系统的重新设计，设计人员日程的重新安排，已经完成的工作可能要重做或者完全抛弃，对其他项目产生影响，硬件需求可能要因此改变等。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相互影响而导致更多问题的出现，需求改变带来的复杂性也可能导致错误。

（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

贫乏或者不规范的文档使得代码维护和修改变得异常艰辛，其结果是带来许多错误。

（7）软件开发工具自身的错误

可视化工具、类库、编译器、脚本工具等，常常会将自身的错误带到应用软件中。

事实上，对于软件来讲，不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。

1.3 软件质量与质量模型

软件质量是软件的生命，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问题。因此，软件质量问题一直都是软件工程的核心问题。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会有不同的理解。常常说：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。这些模模糊糊的语言实在不能算作是软件质量评价，特别不能算作是软件质量科学的定量评价。但是，软件质量乃至任何产品质量，都是一个很