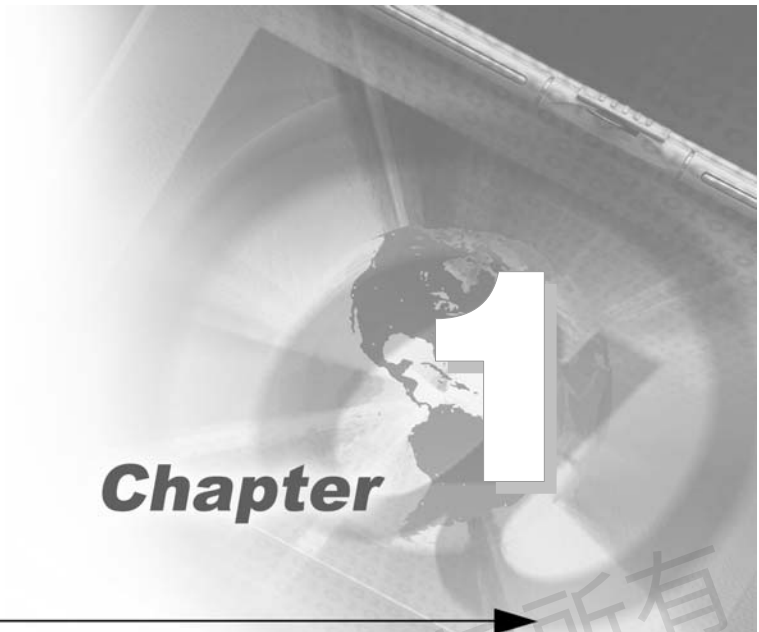



# 第 1 章　软件测试概述



## 本章导学

### 内容提要

本章为软件测试的概要阐述。包括软件测试基本理论知识，包含软件测试产生与发展、软件测试定义、软件开发模式（过程）与测试策略的关联关系、软件测试过程、软件质量及保证体系等主要内容，并简要介绍了软件测试的新技术、新应用及技术发展方向。

### 学习目标

通过本章学习，读者将能正确理解软件测试产生及背景、软件缺陷及故障、软件测试定义等概念，理解软件测试的基本思想与实施策略，能初步认识软件开发与软件测试相辅相成、相互依赖的关系，对软件测试建立概要性、框架性的整体认识和为后续学习软件测试策略、流程与工程方法奠定坚实基础。要求：

- ☒ 正确理解软件测试的背景及定义、软件缺陷、故障和软件失效等概念
- ☒ 正确理解和认识软件测试的目的及测试的意义
- ☒ 正确理解和认识软件测试的基本原理与实施策略
- ☒ 正确理解和认识软件开发过程与软件测试的依存关系
- ☒ 正确理解和认识软件质量的概念及质量保证体系

## 1.1 软件测试的产生与发展

### 1.1.1 软件可靠性问题

软件是人类智力的一种“产品”。这种产品与传统产品的最大区别是无物理实体，是纯粹的逻辑思维的结果，表现为高度的抽象性、形式化和符号化体系。随着计算机技术和应用的迅速发展，软件现已广泛深入人类信息社会活动的各个领域，其软件规模和复杂性与日俱增，千万行的软件系统（产品）已屡见不鲜。因此，其错误产生的概率也在大幅增加，因软件产品中存在的缺陷和故障，所造成的各类损失也在不断增加，甚至带来严重、灾难性的后果。据统计，自软件产生以来，因其错误和故障引发的系统失效、崩溃，与因计算机硬件故障而引发的系统失效、崩溃的比例约为 10∶1。当今全球广泛使用的一些计算机系统软件和应用软件中的缺陷、错误与安全漏洞等，经常被披露或曝光，软件企业不停地发布各种软件产品的修正版本和软件“补丁”的现象已司空见惯。目前，软件的质量问题已成为被所有应用软件和开发软件者广泛而深入关注的焦点。可靠的软件系统应是正确、完整和具有健壮性。人类社会对计算机系统，乃至信息系统的依赖程度越高，对软件的可靠性的要求也就越高。

IEEE 定义的软件可靠性：系统在特定环境下，在给定的时间内无故障运行的概率。

由此，软件可靠性是对软件在设计、开发及所预设的环境下具有特定能力的置信度的一个度量，是衡量软件质量的主要指标。

软件产品的“特殊”性质，使其可靠性必须依赖软件工程的各个环节来保证，经过几十年的经验与总结，提高可靠性的最重要的策略与技术手段是在软件的生命周期中不断进行软件测试。

### 1.1.2 软件缺陷与故障

#### 1. 软件缺陷与故障案例及其意义

尽管软件工程中采取一系列有效措施，不断提高软件的质量，但仍然无法完全避免软件会存在各种各样的缺陷，即软件中存在缺陷几乎不可避免。

软件缺陷或故障（在运行时发生），可依据其可能造成的危害及风险程度，分为极其严重、严重、一般、轻微等不同级别。这里，将通过比较典型的软件缺陷与故障的案例分析，来说明软件缺陷与故障问题所造成的严重后果及灾难。

1）软件缺陷与故障的典型案例

**【事件 1】** 美国迪士尼公司狮子王游戏软件 Bug 事件。这是一起典型的软件兼容性缺陷问题。造成这一严重问题的起因是，这个公司没有在已投入市场的各种 PC 上进行该游戏软件与硬件环境的兼容性的完整测试，使兼容性没有得到保障。虽然该游戏软件在开发者机器的硬件系统上工作正常，但在公众使用的其他各类硬件系统中存在不兼容问题，造成了该软件无法正常运行。用户大量的投诉，致使迪士尼遭受经济与声誉双重重大损失。

**【事件 2】** 美国航天局火星极地飞船登陆事故。1999 年 12 月 3 日，美国航天局火星极地登陆飞船在试图登陆火星表面时突然失踪。负责这一项目的错误修正委员会的专家们观测到了这一幕并分析了事故原因，确定事故可能是由于某一数据位被意外更改而造成灾难性的后果的，并得出造成该事故的问题应在内部测试时就予以解决的结论。事后分析测试发现，机械振动很容易触发飞船的着地触发开关，导致程序设置了错误的数据位，关闭了登陆推进

器燃料开关，切断了燃料供给，使推进器提前停止工作，飞船加速下坠 1800m 直接冲向火星的表面撞成碎片。这一严重事故，损失巨大，但是仅起因于软件的一个小缺陷。

**【事件 3】** 跨世纪的软件“千年虫”缺陷问题。20 世纪末最后几年，全球计算机硬件、软件和应用系统都为 2000 年的时间兼容问题及与此年份相关的其他问题付出代价。据统计，全球仅在金融、保险、军事、科学、商务等领域，对现有程序进行检查、修改，所花费的人力、物力耗资高达几百亿美元。而这些缺陷问题根源在于早期的软件设计，并未考虑跨世纪的 2000 年的时间问题。

**【事件 4】** 美军爱国者导弹防御系统狂炸自己。美军爱国者导弹系统首次应用于海湾战争并屡建功勋，多次成功拦截飞毛腿导弹。但因很小的系统时钟错误积累的延时缺陷，造成了跟踪系统精度的偏差，导致一枚导弹在沙特多哈炸死了 28 名美军士兵。

**【事件 5】** 美国加州监狱计算机程序缺陷致使上千高危犯人被错放。2011 年 5 月，美国加利福尼亚州监察部门表示，由于监狱计算机程序缺陷，信息不完整致使误放近 450 名“高度暴力危险”囚犯和 1000 多名很可能实施毒品、财产犯罪的囚犯被假释出狱。

诸如上述软件缺陷或错误的案例并不仅为这几项，类似的问题数不胜数。现今，全球正在使用的多种软件的缺陷与错误（如系统的安全漏洞等），常常被披露和曝光，不时发布这些软件的修订升级版本或程序补丁，已成为一种常态。

2）软件缺陷与故障定义

从缺陷或故障的案例中已能够认识和理解什么是软件的缺陷和错误了，并观察出软件发生故障或事件的共同特点。软件开发可能未按预期规则或目标要求进行，虽经过测试，但不能保证完全发现和排除了已发现存在的或潜在的错误，甚至有一些是简单而细微的错误。

不论软件存在问题的规模与危害是大还是小，都会产生软件使用上的各种障碍，所以将这些问题统称为软件缺陷。对软件缺陷的精确定义，业界通常认同下列 5 条描述。

（1）软件未达到产品说明书中已标明的功能。

（2）软件出现了产品说明书中指明不会出现的错误。

（3）软件未达到产品说明书中虽未指出但应（隐含）达到的目标。

（4）软件功能超出了产品说明书中指明的范围。

（5）测试者认为软件难以理解、不易使用，或最终用户认为软件使用效果不良。

3）软件缺陷的特征

软件测试理论研究与测试实践都表明，软件缺陷的特征有两个。

（1）软件系统的逻辑性与复杂性等特殊性质决定了其缺陷不易直接被肉眼观察到，即“难以看到”。

（2）即使在运行与使用中发现了缺陷或发生故障，仍不容易找到问题产生的原因，即“看到但难以抓到”。

4）软件缺陷产生原因

软件测试是在对软件需求分析、设计规格说明、编码实现和发布运行之前的最终审定。为何还会存在缺陷呢？研究表明，故障并不一定是由编码过程所引起的，大多数缺陷并非来自编码过程中的错误。因其缺陷很可能在系统概要或详细设计阶段、甚至在需求分析阶段就存在问题而导致故障发生，针对源程序进行的测试所发现的故障根源也可能存在于开发前期。事实上，导致缺陷最大原因在于软件产品的设计文档（设计、规划文本及说明书等）。

在多数情况下，软件产品的设计可能存在着没有文档，或描述不清楚、不准确，或在开发中对产品的需求及产品的功能经常变更，或因开发人员之间没有充分地进行交流与沟通，

或没有组织执行测试的流程等情形。因此，制定软件产品开发计划非常重要，如果产品计划没有制订好，缺陷就会潜伏，软件运行时出现故障问题就在所难免。

根据统计，软件缺陷产生的第二大来源是设计方案，这是实施软件计划的关键环节。

典型的软件缺陷产生原因大致归纳为以下 10 类。

（1）软件需求解释有错误或不明确。

（2）用户需求的定义中存在错误。

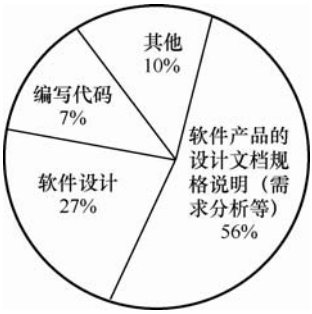



图 1-1　软件缺陷产生原因分布概率

（3）软件需求中记录了错误。

（4）软件设计说明中有错误。

（5）软件编码说明中有错误。

（6）软件程序代码存在错误。

（7）数据输入有错误。

（8）软件测试过程有错误。

（9）软件问题修改不正确或不彻底。

（10）有时，错误结果是因其他软件缺陷而引起或产生的。

图 1-1 所示是软件缺陷产生原因分布概率，软件产品的设计文档规格说明（如需求分析等）占 50%以上。

### 2．软件测试定义

软件测试是发现缺陷的过程。在确定关于软件测试的概念或定义时，通常都会引用《软件测试的艺术》（Myers）一书中的观点。

（1）软件测试是为了发现错误而执行程序的过程。

（2）测试是为了证明程序有错，而不是证明程序没有错误。

（3）一个好的测试用例是它能发现至今未发现的错误。

（4）一个成功的测试是发现了至今未发现错误的测试。

软件测试就是在软件开发和投入运行前的各阶段，对软件的需求分析、设计规格说明和程序编码等过程的阶段性隐藏的错误或缺陷的最终检查，是软件质量保证的关键过程。通常，对软件测试的描述性定义有以下两种。

定义 1：软件测试是为了发现错误而执行程序的过程。

定义 2：软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例（输入数据及其预期的输出结果），并利用这些测试用例运行程序以及发现错误的过程，即执行测试步骤。

## 1.1.3　软件测试的产生与发展

软件测试在 20 世纪 60 年代后正式建立。1961 年，一个简单的软件错误导致了美国大力神洲际导弹助推器的毁灭，致使美国空军强制要求在其后的关键性发射任务中，必须进行独立的测试验证，从而建立了软件验证与确认的方法论，软件测试就此正式产生。

随着软件迅速发展和广泛而深入应用于人类社会与生活各领域，系统规模和复杂性与日俱增，其错误产生的概率大为增加。软件存在的缺陷与故障所造成的损失也在不断发生，一些重要软件系统，如航空航天与高速列车的自动控制软件、银行结算系统与证券交易系统、国家军事防御系统、核电站的安全控制系统，以及涉及公众的交通订票系统、电子商务系统、

生命科学和医疗诊断系统等，因软件系统的质量问题，可能会造成严重损失或带来灾难性的后果。当今，在信息社会的生态系统中，软件质量问题已成为所有开发软件和应用软件者关注的焦点。软件是人脑智能化的一种典型体现，并以思维逻辑的形式呈现为一种“抽象产品”，并且具有生命周期的特征，从而有别于其他科技、生产领域及其产品的形态。软件的这个特性，使其“与生俱来”就可能存在着缺陷，且不易被发现或难以彻底根除。

软件具有“看不见，摸不着”的非有形产品的特征，从而有别于其他传统工业产品的外形特征，因此，软件的质量无法或难以采用传统的工业品的检验方法。在软件开发中的中间（过程）产品及最终产品的质量检验则更为复杂和困难。

软件工程的几十年的发展里程表明，对软件缺陷或错误的检验，预防软件运行发生故障的最有效的措施，就是通过软件测试来发现缺陷或错误，从而控制其质量。

软件测试始终都是软件质量理论研究与工程实践的重要内容。

20 世纪 60 年代，在软件工程建立之初，软件测试是为表明程序的正确性而进行的一项工作。1972 年，在美国北卡罗来纳（North Carolina）大学举行了首次以软件测试为主题的正式学术会议。1973 年，Bill Hetzel 给出软件测试的第一个定义：“软件测试就是对程序能够按预期的要求运行建立起的一种信心。”1975 年，John Good Enough 和 Susan Gerhart 在 IEEE 发表了《测试数据选择的原理》一文，从而使软件测试开始被确定为软件领域的一个新的研究方向与工程实践。1979 年，Glenford Myers 在《软件测试的艺术》一书中提出软件测试的目的是证伪，即“测试是以发现错误为目的而运行的程序或系统的执行过程”。在这个阶段，对软件测试的理解是：“软件测试用于验证软件（产品）能否正确工作并符合要求。”

20 世纪 80 年代后期，全球软件业迅速发展，软件规模越来越大，复杂程度越来越高。如研发的各种操作系统、大型商务软件、航天飞船控制系统、复杂工业流程控制系统等。在此阶段，某些系统的软件开发团队人员达到了几千或上万人的规模，程序也达到了几十万乃至几千万行的数量级。此时，软件的开发成本、效率和质量受到高度的重视，整个过程需要控制。同时，在该阶段，软件测试理论研究和技术应用也得到发展，其最主要的表现就是软件测试定义发生改变。软件测试不再单纯为一个“发现程序缺陷和错误的过程”，而且开始包含对软件质量评价的内容，软件测试被作为软件质量保证的一个重要手段，用以控制、保障和评价软件的质量，并为此制定了软件测试工程与技术的标准。1983 年，Bill Hetzel 在《软件测试完全指南》一书中提出：“测试是以评价一个程序或系统属性为目标的任何一种活动，测试是对软件质量的度量。”与此同时，IEEE 对软件测试的定义是“使用人工或自动的手段来运行或测量软件系统的过程，目的是检验软件系统是否满足规定的要求，并找出与预期结果之间的差异。软件测试是一门需要经过设计、开发和维护等完整阶段的软件工程。”从此，软件测试进入新的发展时期，成为软件领域的专门学科，并开始形成较完整的理论体系与技术方法，测试已具有高度的独立性，测试被正式列入软件工程范畴，并逐渐实现工程化。

进入 20 世纪 90 年代后，软件工程发展迅速，形成各种各样的软件开发模式，同时关于软件质量的研究和技术实践也不断被理论化和工程化，软件开发过程得到规范性和约束性的要求，软件测试与之相辅相成，测试技术规范和软件质量度量建立和逐步形成。1996 年，建立了软件测试能力成熟度模型（Testing Capability Maturity Model，TCMM），其后，软件业界又提出和制定了软件测试支持度量模型（Testability Support Model，TSM）、软件测试成熟度模型（Testing Maturity Model，TMM）和 ISO/IEC 9126 软件质量模型等一系列技术规范及质量标准。与此同时，开始产生与开发了软件测试工具，并开始在测试工程实践中运用测试工具，以辅助手工（人工）测试，加强测试力度和提高测试效率，开始探索软件测试的自动化形式与实施。

21 世纪后，软件测试应用促进了其理论的进一步发展与技术应用的深入。Rick 和 Stefan 在《系统的软件测试》中对软件测试做了进一步的定义："测试是为了度量和提高被测软件的质量，是对被测软件进行工程设计、实施和维护的整个生命周期过程。"新的软件测试理论、测试策略、测试技术、测试领域应用不断涌现，测试活动渗透和深入到各类软件系统中，如基于模型的测试、Web 应用系统的测试、嵌入式系统的软件测试、游戏软件测试等。由此带来测试活动及过程对软件开发技术的相互影响与作用反馈，以及对软件测试的重新评价。

近年来，软件测试与软件开发由相对独立性逐渐开始出现既独立又融合的特性，这是一个显著的变化。开发人员将承担软件测试的责任，同时，测试人员也将更多参与测试代码的开发工作，软件开发与测试的边界变得十分清晰，但过程又融为一体。以敏捷开发模式为代表的新一代软件开发模式，产生和融入了软件开发的新思想、新模式、新策略。极限编程、测试驱动、角色互换、团队模式等，在一流先进软件企业探索和实施，并赢得众多软件开发团队的青睐，不断获得成功。例如，测试驱动开发技术（Test-Driven Development，TDD）就是将测试作为开发工作起点和首要任务的一项新方法，是敏捷开发的一项核心实践和技术，也是一种新的设计方法论。TDD 的基本原则是通过测试来推动整个开发的进行，在开发功能代码之前，先编写单元测试用例的代码，测试代码确定了需要编写什么样的软件代码。TDD 测试驱动的开发技术不仅仅单纯运用于测试工作，而是把需求分析、设计与质量控制量化进行一体化的全过程。

软件测试模型、测试方法和测试服务模式等，也是进入 21 世纪软件测试研究的主要内容与方向。基于测试模型研究与应用，基于云计算、大数据的测试，基于 Web 2.0 软件的测试，基于安全性的测试，基于虚拟技术创建、维护、优化测试环境，乃至测试执行等都成为软件测试领域的新热点、新应用。

对测试质量的衡量已从计算缺陷数量、测试用例数量转到需求覆盖、代码覆盖方面；基于模型的软件测试针对软件中一些常见模型而提出，例如，软件模型分为故障模型、安全漏洞模型、差性能模型、并发故障模型、不良习惯模型、代码国际化模型和诱骗代码模型等。基于模型的测试机理首先提出软件模型，然后通过检测算法对其进行检测。若检测算法是完全的，则能从软件中排除问题。

软件测试的重要性和理论、技术体系和应用的发展，促使软件企业中产生了软件测试的专门组织与机构，组织形式与结构得到规范，测试策略与技术得到更新，自动化测试程度得到提高与完善。与此同时，产生了专门从事软件测试专业工作的机构或企业，使软件测试工作呈现出职业化的特征。2003 年，由德、英、美等国的软件测试工作者分别成立了各自国家的软件测试专业委员会，并在此基础上，成立了国际软件测试专业认证委员会（International Software Testing Qualifications Board，ISTQB），到 2009 年，该组织已扩大到 40 多个国家，成为专门制定软件测试规范标准、开展技术咨询、进行测试专门人才认证与指导的国际性专业组织机构。

软件测试大致经历了软件调试、独立的软件测试、首次被定义、成为专门学科与技术、与软件开发实现融合几个重要阶段。模型软件测试的产生和运用促进了软件测试的学科理论与技术运用的快速发展，新的软件测试理论、新的软件测试策略与方法、新的软件测试技术应用在不断地创新和涌出，软件测试已成为软件领域的专门学科，其测试应用与实践蓬勃发展，软件企业已建立软件测试的专门组织和机构，同时伴随着软件测试工作的专业化和职业化。

软件测试的发展大致经历了如图 1-2 所示的几个重要阶段。

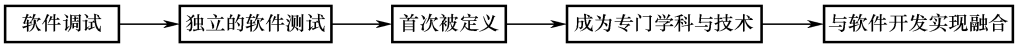



图 1-2　软件测试的发展历程

### 1.1.4　软件测试的发展趋势

#### 1．软件测试领域的动态变化

近十年，软件测试学科研究和技术研发得到快速发展，主要表现为：有了比较完善的针对不同软件系统和软件类型的测试解决方案与测试方法论；研发出的软件测试工具的“智能化”程度越来越高；软件测试的组织形式、测试策略与测试技术不断地发生变化。著名软件专家 Harry Robinson 在 2004 年曾预测，软件测试领域将会发生下列变化。

（1）软件需求工程师、开发工程师将成为软件测试团队成员，并与测试人员合作。

（2）测试方法将日臻完善，Bug 的预防和早期检查将成为测试工具的主流。

（3）通过仿真工具模拟软件正式运行环境进行测试。

（4）对测试用例的更新将变得更为容易（以适应软件需求的变更）。

（5）自动化测试将由机器替代人做更多的工作。

（6）测试执行和测试开发的界限将变得模糊。

（7）对测试质量的衡量将从计算缺陷数量、测试用例数转为需求覆盖、代码覆盖方面。

某些预见，在 2009 年就已实现，如对软件模型的研究取得重大突破，基于模型的软件测试工具应运而生，等等。

#### 2．基于模型的软件测试技术

基于模型的软件测试技术是针对软件中的常见软件模型而提出的一种测试技术。

针对软件模型的分类如下。

（1）故障模型：会引起软件错误或故障的常见模型。

（2）安全漏洞模型：为非法者攻击软件提供的可能性。

（3）差性能模型：软件动态运行时效率低下。

（4）并发故障模型：针对多线程编码。

（5）不良习惯模型：因编码的不良习惯而造成的一些错误。

（6）代码国际化模型：存在于以不同语言进行国际化过程中。

（7）诱骗代码模型：容易引起歧义、迷惑人的编写方式。

基于模型的测试机理：首先提出软件模型，然后通过检测算法进行检测，若检测算法结果符合质量的要求，则能从软件中排除该类模型。基于模型的软件测试工具将能够自动检测软件中的故障，并在对一些大型商业软件和开源软件的测试中发现大量的、以往测试并没有发现的一些软件故障及存在的隐患。例如，IBM AppScan，可自动测试 Web 应用软件系统中的编程安全漏洞。

基于模型的测试技术的优势：

（1）工具自动化程度高，测试效率高，测试过程所需时间较短。

（2）基于模型的测试技术往往能发现其他测试方法难以发现的故障。

基于模型的测试技术存在的不足：

（1）存在误报问题。通常基于模型的测试技术属于静态分析技术，而某些软件故障的确

定需要动态执行的信息，因此对于基于静态分析的工具来说，误报问题不可避免。

（2）存在漏报问题。该问题主要由模型定义和模型检测的算法引起，因为目前对软件模式还无规范、统一与形式化的定义。

（3）模型呈多样性。由于软件模型的多种多样，很难用一种或几种策略方法适应多种模型。

预见软件测试模型、测试方法、测试服务模式在未来将成为软件测试研究的主要内容与方向。

## 1.2 软件测试基础知识与理论

### 1.2.1 软件测试的目的与原则

#### 1．软件测试的目的

软件测试的目的是发现软件存在的故障或缺陷，并借此对软件的质量进行度量。为达此目的，测试活动的目标是尽最大可能找出最多的错误。测试是从软件含有缺陷和故障的假设而进行的，实现这个目标的关键是科学、合理地设计出最能暴露问题的测试用例。

（1）测试是程序执行过程，并限于执行处理有限的测试用例与情形且发现了错误。

（2）检测软件是否满足软件定义的各种需求目标。

（3）执行的测试用例发现了未曾发现的错误，实现成功的测试。

#### 2．软件测试原则

依据软件测试的目的，有以下测试原则。

（1）尽早地和及时地进行测试。测试活动应从软件产品开发的初始阶段就开始。

（2）测试用例要由测试数据与预期结果两个部分组成，并包括测试前置条件或后置条件。

（3）测试根据其需求和风险，可由专业测试人员进行或程序开发者自行检测。

（4）需要严格执行测试计划，并排除测试工作随意性。

（5）充分注意测试中的集群效应，经验表明软件约 80%的错误仅与 20%的程序有关。

（6）应对测试结果做核查，存档测试计划、测试用例、缺陷统计和分析报告等文档，为软件维护提供资料及条件。

### 1.2.2 软件测试的基本原理与特性准则

软件测试发展史已达 40 多年。经过长期实践，总结归纳出了一些测试的基本原理与特性准则，并被业界普遍接受和遵循，对测试的设计、执行和管理均具有工程指导意义。

#### 1．测试的基本原理

**【原理 1】测试可以证明缺陷存在，但不能证明缺陷不存在**

测试可以证明软件系统（产品）是失败的，即说明软件中有缺陷，但测试不能证明软件中没有缺陷。适当的软件测试可以减少测试对象中的隐藏缺陷。即使在测试中没有发现失效，也不能证明其没有缺陷。

**【原理 2】穷尽测试是不可能的**

测试若考虑所有可能的输入值及其组合，并结合所有的前置条件进行穷尽测试是不可能的。在实际测试过程中，软件测试基本上是抽样测试。因此，必须根据风险和优先级，控制测试工作量。

**【原理 3】测试活动应尽早开始**

在软件生命周期中，测试活动应尽早实施，并聚焦于定义的目标上，这样可以尽早地发现缺陷。

**【原理 4】缺陷集群性**

在通常情况下，缺陷并不是平均的，而是集群分布的，大多数的缺陷只存在于测试对象的极小部分中。因此，如在一个地方发现了较多缺陷，通常在附近会有更多的缺陷，这就是所谓的缺陷集群性，也就是经常所说的“80/20 现象”，80%的缺陷集中在 20%的程序模块中。因此，在测试中，应机动灵活地应用这个原理。

**【原理 5】杀虫剂悖论**

若同样的测试用例被一再重复执行，则会减少测试的有效性。先前没有发现的缺陷反复使用同样的测试用例也不会被重新发现。因此，为了维护测试的有效性，战胜这种“抗药性”，应对测试用例进行修正或更新。这样，软件中未被测试过的部分或先前没有被使用过的输入组合会被重新执行，从而发现更多的缺陷。

**【原理 6】测试依赖于测试内容**

测试必须与应用系统的运行环境及使用中固有的风险相适应。因此，对于存在差异的两个系统可以用完全相同的方式进行测试。对于每个软件系统，测试出口准则等应依据其使用的环境分别量体定制。例如，对安全起关键作用的系统与一个电商应用系统所要求的测试是不尽相同的。

**【原理 7】没有发现失效就是有用的系统的说法是一种谬论**

测试找到了导致失效的 Bug 且修正了缺陷，并不能保证整个系统达到了用户的预期要求和需要。因此，没有发现失效就是有用的系统的说法是一种谬论。

**2．测试的特性准则**

（1）对任何软件（产品）系统都存在有限的充分测试集合。

（2）若一个软件系统在一个测试数据（测试用例）集合上的测试是充分的，那么再执行一些测试用例也是充分的，这一特性称为测试的单调性。

（3）即使对软件所有的组成成分都进行了充分测试，也并不能表明整体软件系统的测试已经充分，这一特性称为测试的非复合性。

（4）即使对软件系统整体的测试是充分的，也并不能证明软件系统中各组成成分都已得到了充分测试，这个特性称为测试的非分解性。

（5）软件测试的充分性应与软件需求和软件实现相关。

（6）软件越复杂，测试数据就越多，这一特性称为测试的复杂性。

（7）测试越多，进一步测试所获充分性增长就越少，这一特性称为测试回报递减率。

（8）软件测试的特性准则对测试的设计与执行均具有工程指导意义。

### 1.2.3　软件测试的基本策略

软件具有生命周期的特性。软件测试贯穿整个软件生命周期，因此，测试的基本策略是在其生命周期的每个阶段中确定测试目标、确认测试对象、建立测试生命周期、制定和实施测试策略、选择测试类型和运用测试方法 6 项。

### 1．确定测试目标

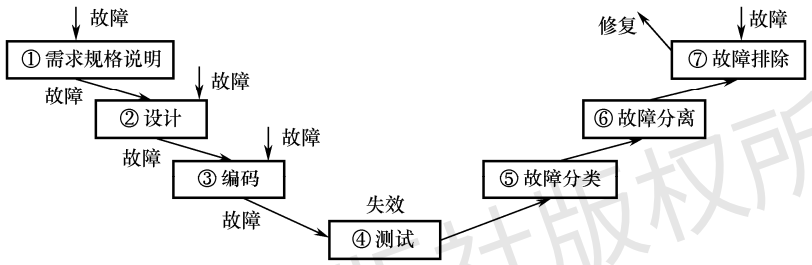
根据软件生命周期划分的几个阶段，以及软件质量包含的各项属性特征，测试需要对每一阶段和每个部分的目标进行确定。

### 2．确认测试对象

软件测试是对程序的测试，并贯穿于软件生命周期整个过程。因此，软件开发过程中产生的需求分析、概要设计、详细设计以及编码等各个阶段所获文档，如需求规格说明、概要设计规格说明、详细设计规格说明以及源程序等都是软件测试的对象。

### 3．建立测试生命周期

软件测试生命周期包括在软件生命周期中。测试生命周期从大的方面看，主要横跨两个历程，分为软件生产阶段的测试历程和软件运行维护阶段的测试活动。软件测试生命周期在软件生产阶段的测试活动及运行维护阶段的测试活动过程如图 1-3 所示。




注：①～③ 可能引入故障或导致其他阶段故障；④ 测试发现了失效；⑤～⑦故障排除。故障排除过程有可能使原本正确执行的程序又出现错误，即排除旧故障、引入了新故障。

图 1-3　软件测试生命周期模型

### 4．制定和实施测试策略

制定和实施测试策略包含以下四项内容。

（1）确定测试由谁执行。在软件产品开发中，通常有开发者和测试者两种角色。开发者通过开发形成产品，如分析、设计、编码调试或文档等可交付物。测试者通过测试检测产品中是否存在缺陷，包括根据特定目的而设计的测试用例、测试过程构造、执行测试和评价测试结果。通常的做法是，开发者负责完成组件级别的测试，而集成级别和系统级别的测试则由独立的测试人员或专门测试机构完成。但也可有其他策略，如在组件级别的测试中，专门测试人员加入。

（2）确定测试什么。测试经验表明，通常表现在程序中的故障不一定由编码引起。它可能由软件详细设计、概要设计阶段，甚至需求分析阶段的问题所致。要排除故障、修正错误必须追溯到前期工作。事实上，软件需求分析、设计和实施阶段是软件故障的主要来源。

（3）确定何时进行测试。确定测试是与开发并行的过程，还是在开发完成某个阶段任务之后的活动或是在整个开发结束后的活动。软件开发经验和事实证明，随着开发过程深入，越是在早期没有进行测试的模块对整个软件的潜在隐患及破坏作用就越明显。

（4）确定怎样进行测试。软件的“规格说明”界定了软件本身应达到的目标，而程序“实现”则是对应各种输入并如何产生输出的一种算法，即规格说明界定软件要做什么，而程序实现表达了软件怎样做。确定怎样进行测试，也就是要根据软件“规格说明”和程序实现的对应关系，确定采用何种测试策略与技术方法，设计并生成有效测试用例，对其进行检验。

5．选择测试类型

软件的特征主要表现在功能性和非功能性上。功能性主要包含适应性、准确性、互操作性、安全性、遵从性四类。非功能性包含可靠性、可用性、有效性、可维护性、可移植性五类，每类又包含若干个子属性。因此，测试设计需要明确测试的类型，并可细分为功能测试、非功能测试、恢复测试和确认测试四类。

1）功能测试（Functional Testing）

功能测试验证系统输入/输出行为的各种功能，其基础是明确功能的需求和系统行为，是基于功能需求的测试，检验系统实现的功能。为检验其正确性和完整性，需采用一系列测试用例进行测试，其中每个测试用例要覆盖功能特定的输入/输出行为检测。功能测试通常采用黑盒测试技术设计测试用例。功能测试是软件测试最重要与最主要的任务。

功能测试常需要进行健壮性测试，检查程序对异常数据的处理情形，以检查被测试对象的异常处理能力，是否存在的缺陷会导致程序瘫痪。因此，健壮性测试也称为功能测试的负面测试（Negative Test）。

2）非功能性测试（Non-Functional Testing）

非功能性测试描述功能行为的属性（或称为系统属性）。非功能性测试包括性能测试、负载测试、安全性测试、可靠性（或稳定性）测试、兼容性测试、可维护性测试以及文档测试等。

关于性能测试、负载测试、压力测试、安全性测试等将在后面章节中阐述。

3）恢复测试（Regression Testing）

这项测试主要检查软件系统的容错能力。当软件出错时，检查能否在指定时间内修正错误并重启系统。恢复测试首先采用各种办法强迫系统失败，然后验证系统能否尽快恢复。对于自动恢复需验证重新初始化、检查点、数据恢复和重新启动等机制的正确性。对人工干预的恢复系统，还需估测平均修复的时间，确定它是否在可接受范围内。

## 1.3 软件开发模式与软件测试

### 1.3.1 软件开发模式

从软件最初的系统分析、系统架构到产品的发布，这个过程称为软件开发模式（也称为开发流程）。采用正确并适宜的开发方法，对开发的组织、管控、质量及进程都很重要。

由于软件的需求、规模与类型的不尽相同，在开发不同软件产品的过程中将会采用不同的开发流程或称模式。目前，主流的软件开发流程方法有很多种，这里选择瀑布模型、快速原型模型、螺旋模型、RUP 模型、IPD 模型、敏捷模型等进行简要分析。

1．瀑布模型

瀑布模型是应用广泛的一种软件开发模型，易于理解和掌握。瀑布模型是将软件生命周期的各项活动规定为按照固定顺序相连的若干阶段性工作，形如瀑布流水，最终得到软件产品。因形如瀑布，故得此名。该模型比较适用于需求稳定并易于准确理解的软件项目开发。瀑布模型如图 1-4 所示。

瀑布模型中各阶段的主要工作及其相应质量控制，如表 1-1 所示。

瀑布模型的优点：易于理解、开发具有阶段性、强调早期的计划及需求分析、基本可确定何时交付产品及进行测试。

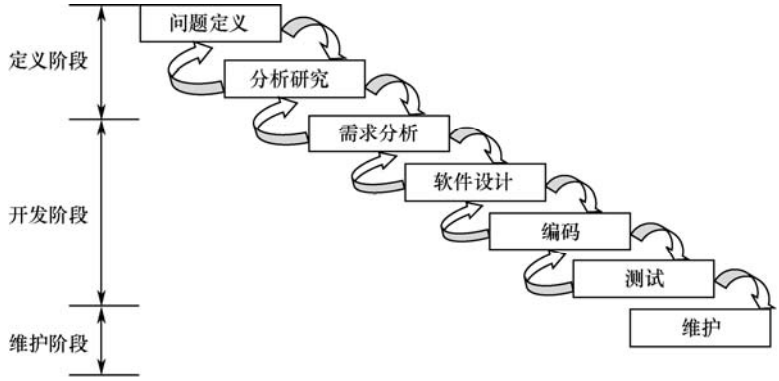



图 1-4　瀑布模型

瀑布模型的缺点：需求调查分析只在最初进行，不能适应需求的新变化；顺序开发流程使开发经验教训不便进行前向反馈；不能反映出开发过程的反复性与迭代特性，无任何类型的风险评估，出现或隐藏的问题直到开发后期才会显露，失去了及早纠正错误或缺陷机会。

**表 1-1　瀑布模型中各阶段的主要工作及其相应质量控制**

| 阶段 | 需求分析 | | | |
|---|---|---|---|---|
| 定义阶段 | 系统需求 | 主要工作 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．调研用户需求及用户环境<br>2．论证项目可行性<br>3．制订项目初步计划 | 1．可行性报告<br>2．项目初步开发计划 | 1．规范工作程序及编写文档<br>2．评审可行性报告及项目初步开发计划 |
| | 需求分析 | 主要工作 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．确定系统运行环境<br>2．建立系统逻辑模式<br>3．确定系统功能及性能要求<br>4．编写需求规格说明、用户手册概要、测试计划<br>5．确认项目开发计划 | 1．需求规格说明<br>2．项目开发计划<br>3．用户手册概要<br>4．测试计划 | 1．需求分析时的成熟工具运用<br>2．规范工作程序及编写文档<br>3．对已完成的 4 种文档进行评审 |
| 开发阶段 | 设计 | 概要设计 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．建立系统总体结构，划分功能模块<br>2．定义一个功能模块接口<br>3．数据库设计（若必要）<br>4．制订组装测试计划 | 1．概要设计说明书<br>2．项目开发计划<br>3．用户手册概要<br>4．测试计划 | 1．进行系统设计时采用先进技术与工具<br>2．编写规范化工作程序及文档<br>3．对已完成文档进行评审 |
| | | 详细设计 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．设计各模块具体实现算法<br>2．确定模块间详细接口<br>3．制定模块测试方案 | 1．详细设计说明书<br>2．模块测试计划 | 1．进行设计时采用先进技术与工具<br>2．规范工作程序及编写文档<br>3．对实现过程及已完成文档评审 |
| | 实现 | 实现编码设计 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．编写程序源代码<br>2．进行模块测试与调试<br>3．编写用户手册 | 1．程序调试报告<br>2．用户手册 | 1．进行系统设计时采用先进技术与工具<br>2．规范工作程序及编写文档<br>3．对已实现过程及完成文档进行评审 |
| | 测试 | 集成测试 | 完成文档 | 完成的文档质量控制手段 |

（续表）

| 阶段 | 需求分析 | | | |
|---|---|---|---|---|
| 开发阶段 | 测试 | 1．执行集成测试计划<br>2．编写集成测试报告 | 1．系统源程序清单<br>2．用户手册 | 1．测试时采用先进技术与工具<br>2．规范工作程序及编写文档<br>3．对测试工作及已完成的文档进行评审 |
| | | 验收测试 | 完成文档 | 完成的文档质量控制手段 |
| | | 1．系统测试（健壮性测试）<br>2．用户手册试用<br>3．编写开发总结报告 | 1．确认测试报告<br>2．用户手册<br>3．开发总结报告 | 1．测试时采用先进技术与工具<br>2．规范工作程序及编写文档<br>3．对测试工作及已完成的文档进行评审 |
| 维护阶段 | 主要工作 | | 完成文档 | 完成的文档质量控制手段 |
| | 1．纠正错误，为完善而进行修改<br>2．对修改进行配置管理<br>3．编写故障报告及修改报告<br>4．修订用户手册 | | 1．故障报告<br>2．修改报告<br>3．修订用户手册 | 1．维护时采用先进工具<br>2．规范工作程序及编写文档<br>3．配置管理<br>4．对维护工作及已完成文档进行评审 |

### 2．快速原型模型

瀑布模型的缺点在于开发过程没有结束前产品不直观。快速原型模型（如图 1-5 所示）改进了这一缺点。通常，根据客户需求在较短时间内解决用户最迫切需要解决的问题，完成一个可演示的产品版本，只实现软件最重要功能（部分）。应用快速原型模型的目的是确定用户的真正需求，使用户针对原型能更明确需求是什么。在得到明确需求后，丢弃原型。因原型开发速度较快，付出不多，也只表达软件最主要的功能，实际应用较多。

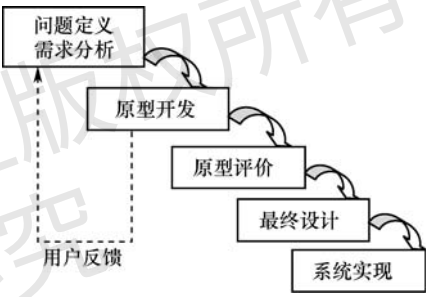



图 1-5　快速原型模型

### 3．螺旋模型

螺旋模型（图 1-6）是瀑布模型与边写边改模型的演化、结合形式，并加入风险评估的一种开发技术。螺旋模型属经典的软件开发方法，包含瀑布模型（分析、设计、开发步骤）、边写边改模型（每阶段盘旋式上升），发现问题早，产品来龙去脉清晰，成本相对低，在测试最初就介入，关注发现并降低项目风险。

该模型被广泛认为是软件开发有效的策略与方法。该模型的主要思想是在开发初始阶段不必详细定义软件细节，而从小规模开始定义重要功能，并尽量实现，然后评测其风险，制订风险控制计划，接受客户的反馈，确定进入下一阶段的开发并重复上述的过程，进行下一个螺旋与反复，再确定下一步项目是否继续，或中止或调整，直到获得最终产品。

每个螺旋包括 5 个步骤：确定目标，选择方案和限制条件；对方案风险进行评估，并解决风险；进行本阶段开发和测试；计划下一阶段；确定进入下一阶段方法。该模型最大优点是随成本增加，风险程度随之降低；具有严格的全过程风险管理，强调各开发阶段的质量，提供机会评估项目，确定是否有价值继续。该模式由于引入严格风险识别、风险分析和风险控制，因此对管理水平提出较高要求，需管理者专注及具备管理经验，并需较多人员、资金与时间的投入。

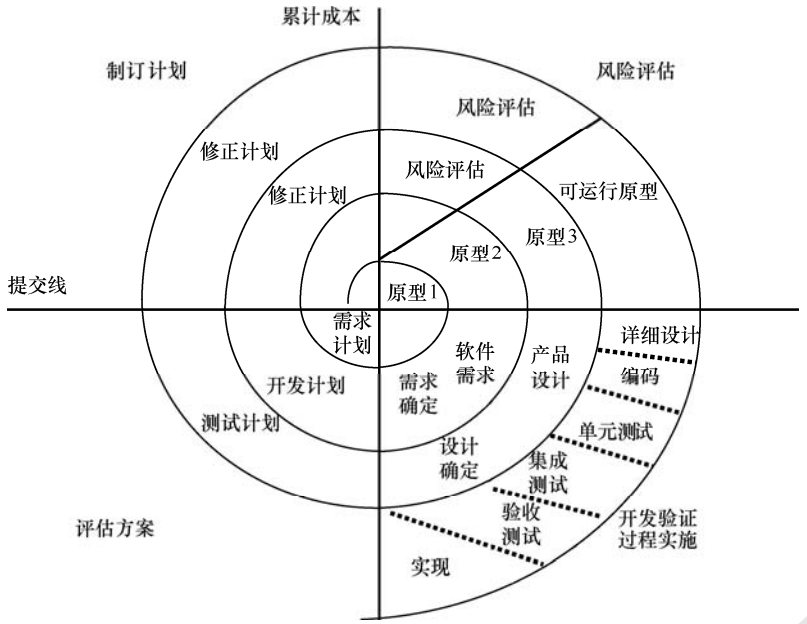



图 1-6　螺旋模型

### 4．RUP 模型

RUP（Rational Unified Process，统一软件过程），是由 IBM Rational 提出的面向对象且基于网络的软件开发方法论，是面向对象软件工程的通用业务流程，是描述一系列相关软件工程的流程，并具有统一的流程构架。RUP 为在开发组织中分配任务和明确职责提供了一种规范的方法，其目标是确保在可预计的时间安排和预算内开发出满足最终用户需求的高品质软件产品。运用 RUP 进行软件的设计开发具有两个轴向：时间轴向为动态，工作流程轴向为静态。在时间轴上，RUP 划分为四个阶段：初始阶段、细化阶段、构造阶段和发布阶段。每个阶段均使用迭代策略。在工作流程轴上，RUP 设计六个核心工作流程与三个核心支撑工作流程。核心工作流程包括业务建模工作流程、需求确定工作流程、分析设计工作流程、设计实现流程、测试工作流程和发布工作流程；核心支撑工作流程包括环境工作流程、项目管理工作流程、配置与变更管理工作流程。如图 1-7 所示。

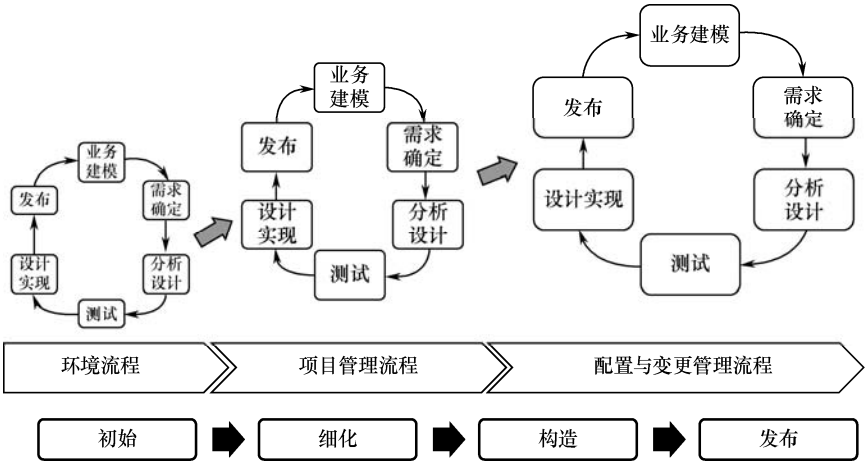



图 1-7　RUP 流程的两个轴向

RUP 汇集了现代软件开发多方面的最佳经验，并为适应各种软件项目及开发组织的需要提供灵活形式，作为一种商业开发模型，它具有非常详细的过程指导与应用模板。

该开发模型比较复杂，因此在模型的运用掌握上需花费较大的成本，并对项目管理提出了较高的要求。

### 5．IPD 模型

IPD（Integrated Product Development，集成产品开发）是由 IBM 提出来的一套集成产品开发流程。它非常适合复杂、大型软件开发项目，尤其是涉及软、硬件结合的开发项目。IPD 从整个产品角度出发，综合考虑了从系统工程、研发（硬件、软件、结构设计、测试、资料开发等）、制造、财务到市场、采购、技术支援等所有流程。

如图 1-8 所示，IPD 有六个阶段（概念阶段、计划阶段、开发阶段、评审（验证）阶段、发布阶段和生命周期阶段）和四个决策评审点（概念阶段决策评审点、计划阶段决策评审点、可获得性决策评审点和生命周期终止决策评审点以及六个技术评审点）。IPD 是一个阶段性模型，具有瀑布模型的因素。该模型通过使用全面且复杂的流程将一个庞大而又复杂的系统进行分解并降低风险。该模型通过流程成本来提高整个产品的质量。由于该模型没有定义如何进行流程回退的机制，因此对于需求经常变动的项目并不适合，并且对小软件项目也不适用。

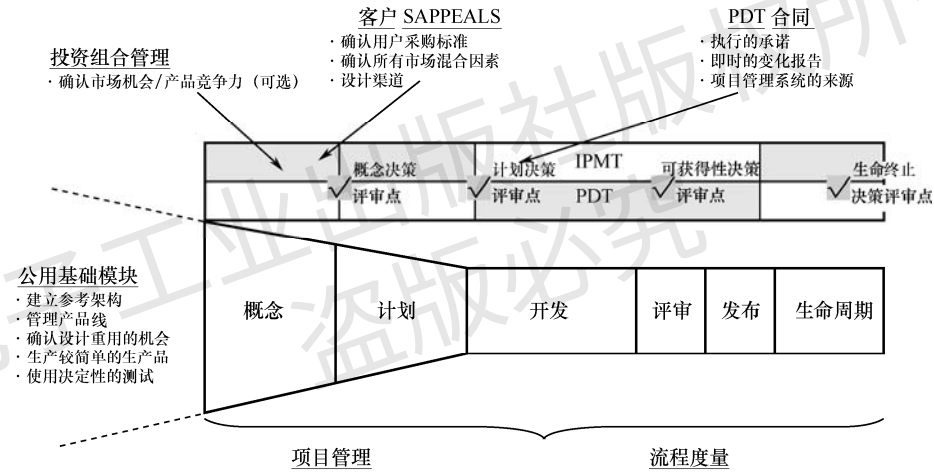



图 1-8　IPD 的六个阶段和四个决策评审点

### 6．敏捷模型

敏捷模型产生于 21 世纪初。它为解决软件企业开发团队陷入不断增长的软件开发过程的“泥潭”和“沉重”负担而概括出能使开发团队具有快速工作和响应变化能力的价值观和原则。敏捷模型不仅是一个开发过程，而是一类过程的统称，这些过程的共性为遵循敏捷原则，提倡简单、灵活与效率，符合敏捷的价值观：交互胜过过程与工具；可运行工作的软件胜过面面俱到的文档；与客户合作胜过合同的谈判；响应变化胜过教条遵循原定计划。

敏捷模型主要包括迭代式增量开发过程（Scrum）、特征驱动软件开发（Feature-Driven Development，FDD）、自适应软件开发（Adaptive Software Development，ASD）、动态系统开发方法（Dynamic Systems Development Method，DSDM），以及很重要的极限编程（eXtreme Programming，XP）方法。

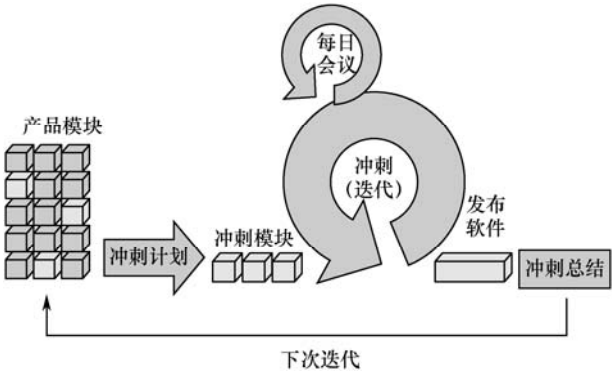



图 1-9　敏捷开发过程原理

（1）Scrum。这是一种迭代和增量的方法，用于管理软件项目（产品）或应用程序开发的过程，如图 1-9 所示。Scrum 的目标是：提升团队的效力；监管团队进程；解决开发过程中的障碍；监管项目过程；使项目的风险最小化。该方法的核心旨在寻求充分发挥面向对象和构件技术的开发方法，是迭代方式面向对象方法的改进。

（2）FDD。这是一套针对中小型软件项目的开发模式，也是一个模型驱动的快速迭代开发过程，它强调简化、实用、易于被开发团队所接受，适用于需求常发生变动的软件项目。

（3）ASD。该方法强调开发方法的适应性，思想来源于复杂系统的混沌理论。ASD 不像其他方法那样有很多具体的实践做法，而更侧重于自适应的重要性和提供最根本的基础，并从更高的组织和管理层面来阐述开发方法为什么要具备自适应性。

（4）DSDM。这是一种快速软件开发方法，应用广泛。该方法倡导以业务为核心，进行快速而有效的系统开发。DSDM 是成功的敏捷开发方法之一，不仅遵循敏捷方法原理，也适合以成熟传统开发方法作为坚实基础的软件企业与组织。

（5）XP。XP 为轻量级、较灵巧的软件技术，同时也是一种严谨、周密的开发方法。XP 技术的基础和价值观是交流、朴素、反馈，任何一个项目都可从强化交流、简单做起、寻求反馈、实事求四个方面入手，并不断进行改善。XP 为近似螺旋模式的开发策略，将复杂过程分解为一个个相对比较简单且较小的周期，通过交流、反馈及其他方法，使开发者与客户都很清楚开发的进度、变化、待解决问题及潜在问题等，并根据实际情况及时调整开发过程。XP 并不完全遵循在软件开发初期就制定很多的文档，以适应开发计划的不断变化和调整，而提倡测试先行或测试驱动的模式，使后期出现 Bug 的概率降至最低。XP 将项目的所有参与者（开发者、测试者及客户）视为同一团队成员，一起工作在开放场所中。

敏捷方法是将开发与测试融为一体，测试以多种不同的方法寻找缺陷和修正。敏捷测试分为手工测试和自动化测试，不同测试扮演不同角色。如人工走查和程序静态测试，分属不同测试类别。根据敏捷原则，确定能用自动化测试事情决不用手工测试，同时做到适于手工测试的内容也不花费高昂成本做自动化测试。

敏捷开发中的测试是整个软件开发的“指引灯”，引领开发过程。在敏捷开发中，测试不完全依赖文档，需要自动地寻找和挖掘更多关于程序的信息来指导测试。在敏捷中，测试人员需主动和开发人员讨论软件需求和设计，研究缺陷出现的原因。敏捷测试需要实施持续测试、不断地回归测试和快速测试。测试为项目组提供各种信息，项目开发过程基于这些信息而做出正确决定。敏捷开发过程的软件测试作用如图 1-10 所示。项目中的测试人员并非质量保证的唯一，整个项目组的每个成员

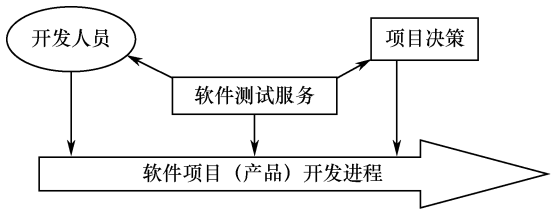



图 1-10　敏捷开发过程的软件测试作用

都对质量负责，而测试人员的主要责任之一是帮助开发人员找到修正的目标。在敏捷开发中，若测试者在团队中采用完全的 XP 方法，则须遵循敏捷原则，调整个人角色，成为真正的敏捷测试者，测试核心工作内容是不断地采取各种方法寻找缺陷，如需着重考虑采用：① 更多采用探索性的测试方法；② 更多采用上下文驱动的测试方法；③ 更多采用敏捷自动化测试原则。

### 1.3.2 软件开发与软件测试

#### 1. 软件开发与软件测试的关系

软件开发模式与软件测试具有密切关系，一般情况下，软件开发与软件测试的关系如图 1-11 所示。

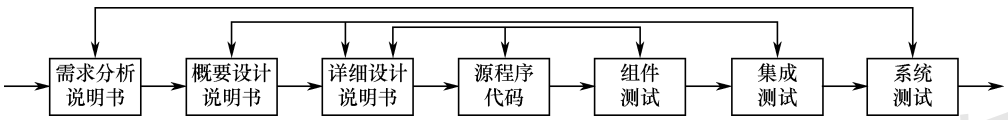



图 1-11 软件开发与软件测试的关系

测试在各阶段的作用如下。

（1）项目规划阶段。负责从单元测试到系统测试的整个测试阶段的监控。

（2）需求分析阶段。确定测试需求分析、制订系统测试计划，评审后成为管理项目。其中，测试需求分析是对产品生命周期中测试所需求的资源、配置、每阶段评判通过的规约；系统测试计划则是依据软件的需求规格说明书，制订测试计划和设计相应的测试用例。

（3）详细设计和概要设计阶段。确保集成测试计划和单元测试计划完成。

（4）编码阶段。由开发人员进行自己负责部分的测试代码。在项目较大时，也可能由测试团队专人进行编码阶段的测试工作。

（5）测试阶段（单元、集成、系统测试）。依据测试代码进行测试，并提交测试状态报告和测试结果报告。

#### 2. 软件测试与开发的并行特性

在需求得到确认并通过评审后，概要设计工作和测试计划制定工作可并行。如系统模块已建立，对各模块详细设计、编码、单元测试等工作又可并行；待每个模块完成后，则进行集成测试和系统测试工作，其流程如图 1-12 所示。

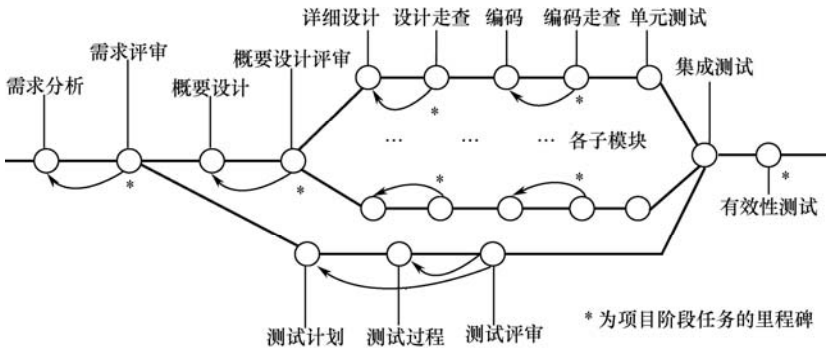



图 1-12 软件测试与软件开发的并行性

### 1.3.3 软件测试模型分析

#### 1. V 模型分析

V 模型是软件开发/测试最经典的模型之一，如图 1-13 所示。V 模型与瀑布模型有共同的特性，可认为是两者的有机融合。IEEE/IEC 12207 定义瀑布模型增强版为通用 V 模型。

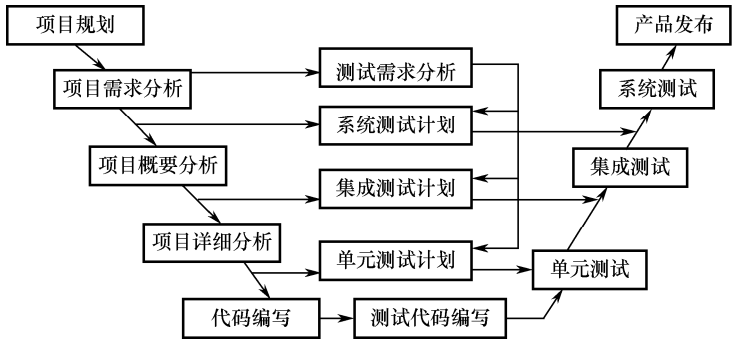



图 1-13　通用 V 模型

V 模型主要反映软件从需求定义到实现与测试活动的关系，强调在整个软件项目生命周期中需要经历的若干开发与测试的对应级别。V 模型从左到右、从上到下，描述开发过程和测试行为。开发行为分布在左边向下分支，测试活动从右边向上，按照测试阶段的顺序排列，明确标明测试过程中存在的不同级别，清晰地描述了测试阶段和开发过程期间各阶段对应关系。

V 模型基于一套必须严格按照一定顺序进行开发的步骤，但很可能没有反映实际的工程过程。V 模型从需求处理开始，提示开发者对各开发阶段已得到的内容进行测试，但没有规定要获取多少。如没有任何需求资料，开发人员就较难以知道该做什么。该模型体现“尽早地和不断地进行软件测试”原则。在软件需求和设计阶段，测试活动遵循软件验证与确认的原则。

V 模型的不足是需求、设计、编码活动被视为串行，同时，测试和开发活动也保持线性的前后关系，只有上一阶段完成才开始下一阶段活动，因此该模式难于支持迭代方式和不适合在开发过程中做变更调整。在 V 模型中增加软件各开发阶段应同步进行的测试，演化为基于 V&V 原理的 W 模型，如图 1-14 所示。W 模型是 V 模型的发展。该模型体现“尽早地和不断地进行软件测试”原则。从模型结构不难看出，开发是“V”，测试是与此并行的“V”，形成双“V”，即“W”结构。在软件需求和设计阶段的测试活动遵循 IEEE 1012—1998《软件验证与确认（V&V）》的原则。

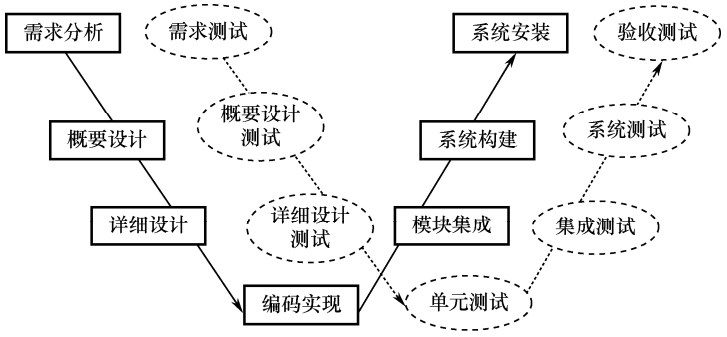



图 1-14　W 模型

相对于 V 模型，强调测试伴随整个软件开发周期，而且测试对象不仅是程序，对需求、功能和设计同样要进行测试（或评审），测试与开发同步，从而有利于尽早发现软件的潜在问题。同样，W 模型不支持迭代模型和不适合开发过程中的变更调整。

### 2．X 模型分析

X 模型是对测试过程模式进行重组，形成“X”。X 模型的示意图如图 1-15 所示。X 代表未知，包括探索性测试的特征。

X 模型左边描述的是针对单独程序片段所进行的相互分离的编码和测试，此后将进行频繁的交接，通过集成最终合成为可执行的程序（图 1-15 右上半部分），对这些可执行程序还需要进行测试。对已通过集成测试的成品进行封版，并提交用户，也可作为更大规模和范围内集成的一部分。多根并行的曲线表示变更可在各部分发生。

X 模型还定位了探索性测试（图 1-15 右下方）。这是不进行事先计划的特殊类型测试，如“这样测试结果怎样？”这种方法帮助有经验的测试人员在测试计划之前，探索发现更多的缺陷。

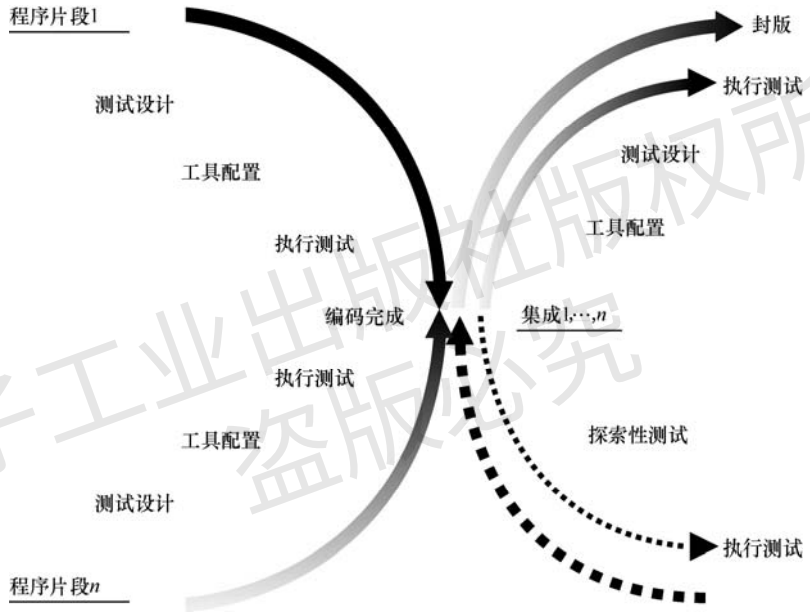



图 1-15　X 模型的示意图

模型和项目计划不同。模型不描述每个项目具体细节，模型只对项目进行指导和支持。在执行测试前进行测试设计，X 模型包含了测试设计步骤，而 V 模型没有这一特征。

在 X 模型和其他模型中都需要有足够的需求，并至少进行一次发布。因此，V 模型的一个特点是明确的需求角色确认，而 X 模型没这个机制，这是 X 模型的不足之处。

具有可伸缩性的行为期望结合进 X 模型，以使模型并不要求在创建可执行程序（图 1-15 右上方）的一个组成部分之前进行集成测试。图 1-15 中左侧行为表明了 X 模型对每一个程序片段都要进行单元测试。

### 3．前置测试模型分析

前置测试模型是将测试和开发紧密结合的模型，代表测试的新思想和新理念。该模型提供灵活方式，可使软件开发加快进度。前置测试为需要使用测试技术的开发人员、测试人员、

项目经理和用户带来不同于传统方法的内在价值，它用较低成本及早地发现错误，并且充分强调测试对确保系统高质量的重要意义。前置测试模型如图 1-16 所示，其模型要点如下。

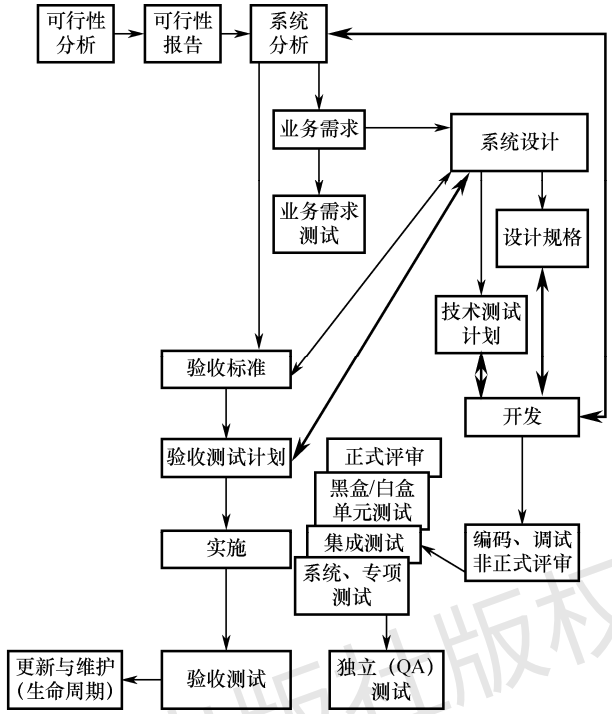



图 1-16　前置测试模型

（1）开发和测试相结合。前置测试模型将开发和测试生命周期整合在一起，标识了项目生命周期从开始到结束之间的关键行为。若其中某些行为没有得到很好执行，那么项目成功可能性会因此而降低。

（2）对每一个交付内容进行测试。每一个交付开发的结果都必须通过测试。测试源程序代码不是唯一需要，包括可行性报告、业务需求说明，系统设计文档等都是测试对象。这与 V 模型中开发和测试的对应关系一致，并在其基础上有所扩展，更为明确。

（3）前置测试模型包括两项测试计划技术。第一项技术是开发基于需求的测试用例。为以后提交的程序测试做好初始化准备，也为验证需求是否可测试。这些测试可交由用户进行验收测试或由开发部门做某些技术测试。需求的可测试性是需求的最基本属性之一，必要时可为每一需求编写测试用例。第二项技术是定义验收标准。在接受交付的系统之前，用户需要用验收标准进行验证。验收标准并不仅是定义需求，还应在前置测试之前进行定义，以揭示某些需求是否正确，或某些需求被忽略了。系统设计在投入编码实现之前也必须测试，以确保其正确性和完整性。

（4）在设计阶段进行计划和测试设计。设计阶段是实施测试计划和测试设计的最好时机。前置测试模型将验收测试中所包含的 3 种成分的 2 种都与业务需求定义相联系，即定义基于需求的测试和定义验收标准，但第 3 种则需要等到系统设计完成。因为验收测试计划由针对按设计实现的系统进行，由一些明确操作的定义所组成，这些定义包括如何判断验收标准已经达到，以及基于需求的测试已被认为成功完成。

（5）前置测试增加了静态审查，以及独立的 QA（Quality Assurance，质量保证）测试。

QA 测试通常跟随在系统测试之后，从技术部门的意见和用户的预期方面出发进行最后检查，包括负载测试、安全性测试、可用性测试等。

（6）测试和开发结合在一起。前置测试将测试执行和开发结合在一起，并在开发阶段以编码—测试—编码—测试方式体现，即程序片段一旦编写完成，就立即测试。通常，首先进行单元测试；其次，也参考 X 模型，对一个程序片段也需进行相关集成测试，有时还需进行特殊测试。对一特定程序片段，其测试顺序可按 V 模型规定，其中还可交织一些程序片段开发，而无须一定按阶段进行完全隔离。

（7）验收测试和技术测试应保持相互独立。前置测试模型提倡验收测试和技术测试沿两条不同路线进行。每条路线分别验证系统能否按预期的设想进行正常工作。这样，当单独设计好的验收测试完成了系统的验证，即可确信这是一个正确的系统。验收测试既可在实施阶段第一步执行，也可在开发阶段最后一步执行。

（8）反复交替的开发与测试。项目中会发生变更。该测试模型对反复和交替进行非常明确的描述。如需重新访问前一阶段内容，或跟踪并纠正以前提交内容，修复错误，排除多余成分，增加新发现功能等，开发和测试需要反复交替执行。

## 1.4 软件质量及其保证

### 1.4.1 软件质量体系

#### 1．质量与质量管理

GB/T 6583—1994 idt（等同于）ISO 8402：1994 定义质量为：“反映实体满足明确和隐含需要的能力和特性综合”。这里实体指产品、活动、过程、组织的体系等。因此，质量“是一组固有特性满足要求的程度”。要确保产品质量，必须保证有生产过程的质量和组织体系的质量等实体的质量。

ISO 9000 的一个重要科学依据是“质量生存于全部的生产过程中”。它具体体现在事前计划、严格按计划实施、事后检查、总结分析并采取改进措施的循环方式上。

所谓质量管理就是组织在产品生产中的质量策划、质量控制、质量保证和质量改进等与质量有关的相互协调的活动，有下列内容。

（1）质量管理体系。ISO 9000:2000 定义的质量管理：在质量方面指挥和控制组织的协调的活动。这些活动构成质量管理体系，是确定质量方针、目标和职责，指导和控制组织所有与质量有关的相互协调的活动，通常指质量策划、质量控制、质量保证和质量改进。

（2）质量管理运作实体，主要为 4 个部分：① 组织结构；② 程序；③ 过程；④ 资源。

（3）ISO 8402：1994 定义的质量策划：确定质量以及采用质量管理体系要素和要求的活动。它包括：① 产品策划；② 质量管理体系管理和运作策划；③ 编制质量计划。

（4）质量控制。监督过程的质量，排除质量环节中影响产品质量的可能因素，使过程的结果满足规定的质量要求。质量控制的方法通过适宜、有效措施和手段监督过程。

（5）质量保证。为了提供足够的信任证据，证明组织有关的各类实体有能力满足质量要求，所实施的有计划、系统的活动。质量保证两个目的：内部质量保证和外部质量保证。前者是在组织内部，后者是向客户或第三方认证提供信任的保证。

（6）质量改进。质量管理是动态、可持续改进的体系。质量改进目的是向组织的所有

受益者提供更多的收益所采用的提高质量过程和效率的各种措施。

现代质量管理已从单纯对产品质量检验拓展到对产品形成过程控制，控制策略也从静态发展到动态与持续的过程改进，其核心思想为对过程的策划、控制和过程能力持续改进。

### 2．软件质量管理

ANSI/IEEE Std 729—1983 定义的软件质量：与软件产品满足规定的和隐含的需求的能力有关的特征或特征的全体。这个定义实质上反映了三个方面。

（1）软件需求是度量软件质量的基础。

（2）在各种标准中定义开发准则，指导软件开发要使用工程化的方法。

（3）软件需求中常有一些隐含需求未明确提出。这表明，软件只满足精确定义的需求，而没有满足隐含的需求，有可能质量得不到保证。

软件质量是软件产品特性可满足用户的功能、性能需求的能力。

ISO/IEC 9126 是软件质量的完整定义标准，其质量包括六个部分：功能性、可靠性、可用性、有效性、可维护性和可移植性。每个部分的质量为软件属性的各种标准度量的组合。软件质量模型与测试组成如图 1-17 所示。

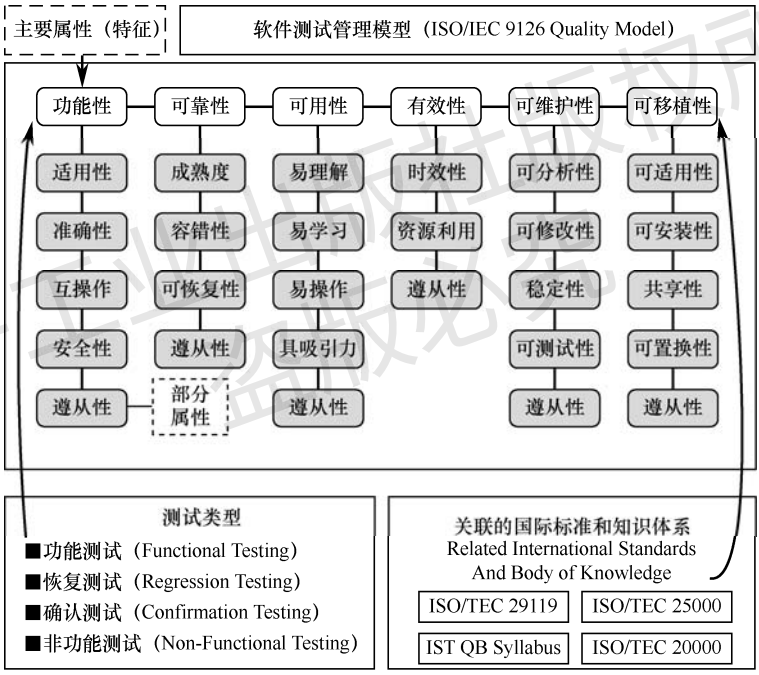



图 1-17　软件质量模型与测试组成

要确保软件质量，必须保证有软件生产过程的质量和软件组织体系的质量。所谓软件质量管理就是软件组织在软件生产中的质量策划、质量控制、质量保证和质量改进等与质量相关的相互协调的活动。与其他质量管理的概念一样，软件质量管理具有质量管理的特征和所有属性。

1）软件质量策划内容

软件质量策划内容如下。

（1）确定组织。为建立软件组织的质量管理体系所做的基础准备。

（2）确定组织的质量管理体系目标。通常以 CMM 及 ISO 9126 作为其质量管理体系的符合性标准或模型。

（3）标识和定义组织质量过程。对组织的质量过程进行策划，确定过程的资源、主要影响因素、作业程序和规程、过程启动条件和过程执行结构的规范等。过程策划不仅是其策划过程本身的质量因素，还要考虑过程间的关系与相互影响。过程策划是质量策划最艰巨、关键的任务。过程策划结果是建立组织的标准软件过程。

（4）标识产品质量特性。建立起目标、质量要求和约束条件。产品策划的关键是确定产品的特性和类型，遵循过程策划结果，定义具体产品或项目质量过程。用项目定义软件过程描述该策划，以及策划质量改进的计划、方法和途径。

2）软件组织的质量过程

软件组织的质量过程通常由软件工程过程与组织支持过程构成。

（1）软件工程过程。软件生命周期中的活动，包括软件需求分析、软件设计、软件编码、软件测试、交付、安装使用与软件维护。将软件工程的基本活动按照生命周期模型组织起来，就构成软件的基本过程。一个组织的软件过程策划包括两个阶段：组织标准生产过程策划阶段与项目策划阶段。

在 CMM 中定义了三个关键区域来实现这两级的过程策划。组织过程定义，其主要任务是识别和确定组织的质量过程，将组织比较成熟的软件过程、过程自愿要求、过程程序、过程产品要求等文档化并形成制度，贯穿于整个组织，以改进所有项目的过程性能。

项目策划阶段：其目的是为具体软件项目的开发、检查活动制订合理计划。项目策划本身就是质量策划的一项活动过程，包含确定项目开发的主要活动及活动间的关系、制定项目开发进度、配备相关资源、设定合适的检查点及检查方式。

（2）组织支持过程。是为保证软件工程过程实施和检查而建立的一组公共支持过程，主要包括管理过程与支持过程。管理过程包括评审、检查、文档管理、不合格品管理、配置管理、内部质量审核和管理评审，支持过程通常不属于软件生命周期的活动。

3）软件质量控制

软件质量控制的主要目标是按照质量策划要求，对质量过程进行监督和控制。它主要有 5 项内容。

（1）组织中与质量活动有关人员依照职责分工进行的质量活动。

（2）所有质量活动依照策划的方法、途径和时间有序地进行。

（3）对关键过程和特殊过程实施适当过程控制方法。

（4）所有质量活动的记录完整且真实保存，以供统计分析时使用。

（5）软件质量控制主要涉及的技术：配置管理和过程流程管理。

4）软件质量度量

软件质量度量分为产品质量度量和过程质量度量。

依赖于具体的产品质量标准，通过测试获得产品质量特性有关数据，辅助适当统计技术确定产品是否满足规定的质量要求。软件的度量包含复杂性度量、可靠性度量等，目前主要采用实用统计方法，如测试中的 Bug 发生率、千行程序误码率等。

软件质量过程度量，目前不如其他制造业成熟。这是因为软件被看成高智力、高创造性的工作。随着软件工程理论发展与成熟，软件业越来越向制造业靠近，对过程度量的成熟不是表现在简单复制等再生产环节的成熟上，而是整个软件过程的成熟上，目前对软件产品的设计、开发、测试、评审等都开展过程的度量；在软件质量过程管理中，对软件的验证通常包括对各层级设计的评审、检查及各阶段测试。对过程验证是对过程数据的评估与审核。

5）软件质量改进

质量改进是现代质量管理的必然，ISO 9000 规定组织定期进行内审和管理评审，采取有效的纠正和预防措施，保持组织的质量方针与目标的持续发展。

软件质量策划对软件过程质量改进有具体的要求，如对时间、资源、计划、目标等进行策划与准备。具体过程改进活动包括以下几项。

（1）质量度量与审核。通常包括软件可靠性度量、软件复杂性度量、软件缺陷度量、软件规模性度量。度量活动涉及需求分析、实现、测试、软件维护，也包括从代码实现到各种评审的内容。

（2）纠正和预防措施。这是质量改进的重要手段，必须认真策划和执行。“纠正”是针对不合格情况的本身，而纠错是消除实际的不合格原因。预防措施与纠正措施不同，预防措施的目的是消除潜在的不合格原因，防止不合格的发生；纠错的目的在于防止不合格再次发生。通过对各种审核报告、评审报告、服务报告、质量记录、客户投诉等信息的分析，来寻求不合格的潜在原因，并针对这些原因采取相应的预防措施。

（3）管理评审。实现持续改进的重要环节，需定期对质量管理体系现状和适应性进行评估。

### 1.4.2 软件测试成熟度

1．软件能力成熟度模型（Capability Maturity Model，CMM）

这是软件业标准模型，用来定义和评价软件企业开发过程成熟度，并提供如何做才能提高软件质量的指导。CMM 将软件组织过程能力成熟度级别分为 5 个级别，每一个级别定义一组过程能力目标，并描述要达到这些目标应采取的活动。依据 CMM，软件开发组织能大幅度提高按计划、高效率、低成本地提交有质量保证的软件产品的能力。

1）CMM 的基本过程

IEEE 对过程的定义：“为达到目的而执行的所有步骤的系列”。软件开发和维护软件及其相关产品的一组活动、方法、实践和改革称为软件过程。软件及其相关产品是指项目计划、设计文档、代码、测试用例、用户手册等。

软件过程结构是对组织标准软件过程的一种高级别描述，明确组织标准软件过程内部的过程元素之间的顺序、接口、内部关系，以及与外部过程之间的接口和依赖关系。

软件过程元素是描述软件过程的基本元素。每一个过程元素包含一组定义的、有限的、封闭的相关任务。过程元素的描述应是一个可填充的模板、可组合的片段、可求精的抽象说明，或可修改或只使用不能修改。

软件过程定义当一个组织的标准软件过程达到高一级别 CMM 时，就定义了一个组织软件过程管理的基础。在企业一级，以正式方式描述、管理、控制和改进组织的标准软件过程；在项目一级，则强调项目定义软件过程的可用性和项目的附加值。对过程应像产品一样进行开发和维护。

2）CMM 的分级结构和主要特征

CMM 描述和分析了软件过程能力的发展程度，确立了一个软件过程成熟程度的分级标准，如图 1-18 所示。一方面，软件组织可利用它评估自身当前过程成熟程度，并以此提出严格的软件质量标准和过程改进方法策略，不断达到更高成熟度；另一方面，CMM 标准也作为用户对软件组织的一种评价，使之在选择软件开发厂商时不再盲目和无把握。

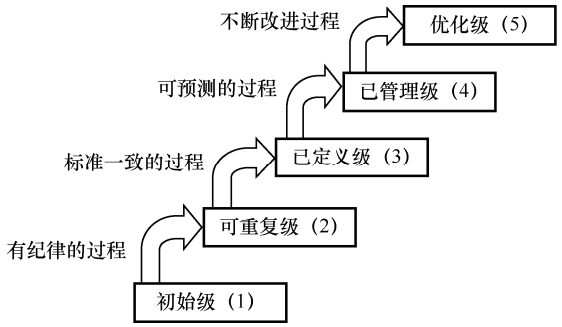



图 1-18　软件过程成熟度的 5 个等级

分级结构和主要特征如下。

（1）初始级。该级特点是软件过程无秩序，有时甚至混乱。软件过程定义几乎无章法和步骤可循状态，软件产品所取得成功往往依赖个别人的努力和机遇。

（2）可重复级。已建立基本项目管理过程，可用于对成本、进度和功能特性进行跟踪。对类似的应用项目，有章可循并能重复以往所取得的成功。

（3）已定义级。用于管理的、工程的软件过程均已实现文档化、标准化，并形成整个软件组织的标准软件过程，全部项目均已采用与实际情况相吻合、适当修改的标准软件过程进行。

（4）已管理级。软件过程和产品质量有详细的度量标准。软件过程和产品质量得到了定量的认证和控制。

（5）优化级。通过对来自过程、新概念和新技术等方面的各种有用信息的定量分析，能不断地、持续性地对过程改进。CMM 的 5 级划分给出了软件组织开展实施活动的应用范畴。

第一级是基础，大部分准备按照 CMM 体系进化的软件企业都自然处于该级别，并从该起点向第二级迈进。除第一级外，每一级都设定了上一级目标，如达到该成熟级别，自然向上一级迈进。从第二级起，每一低级别实现都是高级别实现过渡到上一级别的成熟阶段。分级方式使 CMM 具有可操作性。

3）CMM 分级结构特点

判定成熟度等级有关的组成部分处于模型的顶层，为成熟度等级（Maturity Levels）、关键过程域与各关键过程域目标。除第一级外，CMM 每一级按完全相同的内部结构构成。成熟度等级为顶层，不同的成熟度等级反映了软件组织的软件过程能力和该组织可能实现预期的程度。在每个成熟度级别（一级除外）中包含了实现这一级目标的若干关键过程域 KPA（Key Practice Areas）。CMM 根据过程改进规律，约定公共特性和关键实践内容。每一级每个关键过程域包含若干关键实践 KP（Key Practice）。无论哪个 KPA，都统一按照 5 个公共特性进行组织，即每一个 KPA 都包含 5 类 KP。这使整个软件过程改进工作自上而下形成有规律的步骤，关键域、目标、公共特性和关键实践。

4）关键过程域及目标

所谓关键过程域是指一系列相互关联的操作活动，这些活动反映软件组织改进过程必须集中力量改进的几个方面，即关键过程域包含了达到某个成熟程度级别时所必须满足的条件。CMM 每个成熟度级别规定不同关键过程域，软件组织如希望达到某一成熟度级别，就必须完全满足关键过程域所规定的不同要求，即满足每一个关键过程域的目标。CMM 共有 18 个关键过程域，分布在第 2～5 级中，在 CMM 实践中起至关重要的作用。若从管理、组织和工程进行划分，KPA 可归结成如图 1-19 所示的情形。

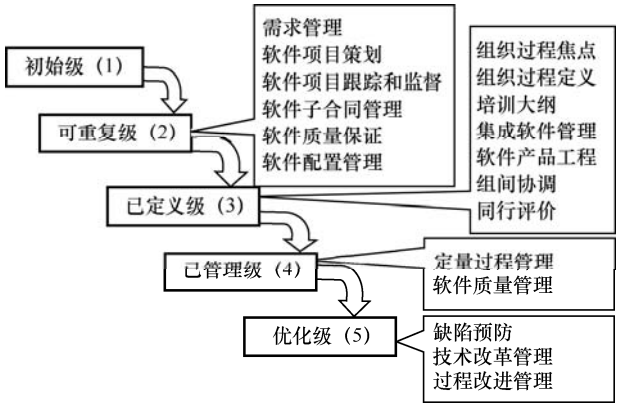



图 1-19　依据成熟度等级排列的关键过程域

关键过程域目标。CMM 中的目标是指某个关键过程域中的关键实践，它表示每一个关键过程域的范围、边界和意图。采用目标可判断一个组织（企业）或项目是否有效地实施了某关键过程域所规定的内容，即目标是检查关键过程域是否满足要求的一个指标。

5）公共特性

为完成关键过程域中的实践活动，CMM 将其活动分为具有公共特性的 5 个部分，这些特性有效地指定了一个关键区域的实现范围、结构要求和实施内容。对于关键过程域的详细实施活动，基本按照这 5 个公共特性进行描述。

执行约定描述的是一个组织为了保证过程得以建立和持续发挥作用所必须采取的行动。

执行能力描述的是在软件每个项目或整个组织必须达到的前提条件。具体执行的能力大小一般与资源、组织机构以及员工训练有关。

实施活动描述的是为实现一个软件过程关键区域所规定步骤而开展的工作，和对该工作进行跟踪，以及在必要时进行改进的措施。

度量和分析描述的是度量的基本规则，以确定、改进和控制过程的状态。度量和分析包括一些度量实例，通过这些实例知道如何确定操作活动的状态和效果。

验收实施描述是判断所开发的实践活动与确立的过程是否遵循了已制定的步骤。验证活动实施可通过管理和软件质量保证进行核查。

6）关键实践

关键实践指一些主要实践活动。每个关键过程域最终由关键实践所组成，通过实现这些关键实践来达到关键过程域的目标。通常，关键实践描述应该“做什么”，并未规定“如何”达到目标，具体的操作方法与步骤须由项目组织自行解决。

CMM 通过内部结构的规范，使软件组织能制定方针、政策、标准，并根据自身的行为建立软件过程，以提高软件过程的成熟度。

### 2．软件测试成熟度模型（Test Maturity Model，TMMi）

软件测试成熟度模型基于 CMMi（软件能力成熟度集成）原则架构，是集成的软件测试成熟度框架，它由 5 个成熟的测试过程级别构成，在每一测试层级中都包含若干个过程域，并由若干过程组成与定义相关联的活动。TMMi 在全球软件业中具有先进指导作用和在测试工程中的广泛应用性。通过软件企业实施 TMMi 规范程度（级别），能够证实该企业对软件测试成熟度水平的衡量和对软件质量的评价能力。TMMi 结构如图 1-20 所示，各测试等级的内涵如下。

（1）初始级。该级没有过程域，主要进行的是缺陷的探索性测试。

（2）可管理级。过程域：由测试方针与策略、测试计划制定、测试监测与控制、测试设计与执行、测试环境五个部分组成。在该级中主要进行软件质量的测试。

（3）重定义级（集成级）。过程域：由测试组织、测试培训、测试生命周期与整合、非功能性测试、同行审查五个部分组成。在该级中主要进行需求确认测试。

（4）管理改进级（可管理与测量级）。过程域：由测试衡量、软件质量评估、高级同行评审三个部分组成。在该级中主要进行质量控制测试。

（5）优化级。过程域：由缺陷预防、测试过程优化和质量控制三个部分组成。在该级中，测试的主要特征是进行持续的测试改进。

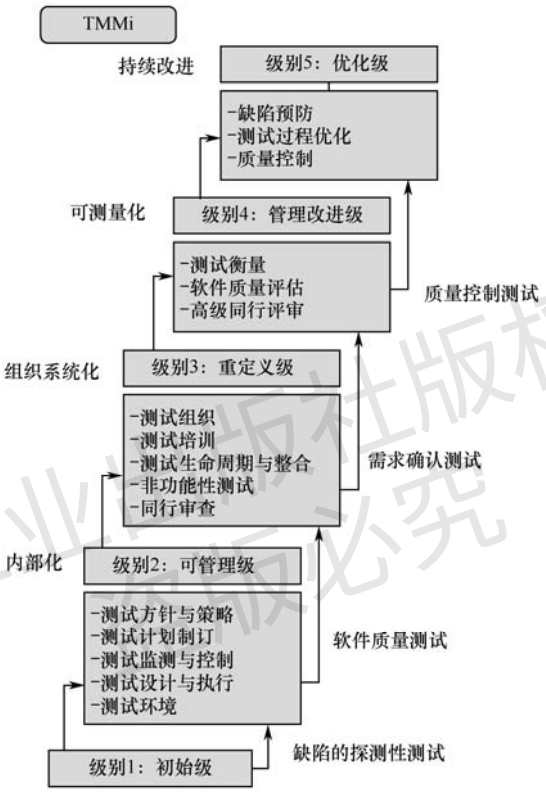



图 1-20　TMMi 结构

## 专业术语

**错误、失效和故障**　软件失效（failure）是指不完全符合给定的需求，实施及结果（actual result）或行为（在执行测试时观察到的）与期望结果（expected result）或行为（在软件规格说明和需求中定义的）之间的偏差。

导致软件失效的是其自身错误（fault）、缺陷（defect）或故障（bug），均在软件开发或更改后存在，并只在软件被执行时这些故障才会以失效方式暴露出来。失效根源在软件中某个故障。这种故障也称为缺陷或内部错误。程序中的代码错误和代码疏漏都为故障。

**缺陷屏蔽**（defect masking）　故障被程序其他部分某个或某些故障所掩盖，只在修正了屏蔽它的故障，相应失效才能显现出来，这也说明修正故障有可能产生副作用。

**测试与调试**　查找失效的过程称为测试。测试的目标是发现系统的失效，证明缺陷的存

在。调试是对缺陷的定位与修正。调试一般不能解决软件逻辑正确性和软件功能、性能上的问题，调试的目标是改错，而不是寻找软件的缺陷。

**测试用例**　是为特定目的而设计的一组测试条件、输入测试对象的预期输出或预期行为的定义，是执行测试的最小实体，是一项活动，其结果被观察与记录。

**测试套件**　由一个或多个测试用例组成的集合体。

**测试场景**　通过将一个测试用例的输出结果作为另一个测试用例的输入条件，可将多个测试用例组合成为测试场景。

**软件质量特征**　由 ISO/IEC 9126 定义。功能性（functionality）、可靠性（reliability）、可用性（usability）、效率（efficiency）、可维护性（maintainability）以及可移植性（portability）都属软件质量的范畴。测试是衡量软件质量的有效手段，质量特征须在测试计划中界定以判断软件产品的总体质量。

**质量保证**（Quality Assurance，QA）　QA 是不可或缺的质量管理手段。软件测试通常为“事后检查”，只能保证尽量暴露软件缺陷。要真正有效控制和提高软件质量，需从设计阶段考虑，从发现的缺陷中溯源，分析产生原因，制定纠正和预防措施，确保下次不出现相同错误。

**质量控制**（Quality Control，QC）　QC 也是不可或缺的质量管理手段。测试是 QC 的一种手段，为保证质量提供各种有效数据。QA 与 QC 既有共同点，也有区别。两者都寻找错误，但 QC 查找的是产品中的错误，QA 查找的是过程中的错误，两者的目标都是对质量进行管理。

## 本 章 小 结

本章主要的知识点如下。

（1）软件及其特性。软件是人类智力的体现，是逻辑思维的产品，软件为非有形产品。软件具有不便与无法采用直接方法进行质量检验的特性。

（2）软件测试目的。以发现故障或缺陷为目的，尽最大可能找出最多错误，执行测试用例并发现错误，检查软件是否满足定义的需求。

（3）软件测试产生原因。与日俱增的软件规模与复杂性使其产生错误的概率增加，关键软件的质量问题可造成严重损失或灾难，已成为人们开发和应用软件的关注角度。

（4）软件测试概念。对软件必须进行独立的测试与验证；软件测试是对程序能按预期要求运行建立起的一种信心，测试目的是证伪：测试是以发现错误为目的而运行的程序或系统的执行过程；软件测试发现程序缺陷与错误，并对软件质量进行评价，是对缺陷与故障最有效的检验与预防措施。

（5）软件测试定义。测试是为了度量和提高被测软件的质量，是对被测软件进行工程设计、实施和维护的整个生命周期过程。

（6）软件缺陷的特征。软件逻辑性与复杂性决定其缺陷不易直接被肉眼发现，即“难以看到”；即使在运行与使用中发现缺陷或故障，仍不易找到其产生原因，即“难以抓到”。

（7）软件缺陷产生原因。分析产生软件缺陷的原因及比例。

## 习题与作业

### 一、选择题

1.【单选】以下关于软件测试目的的描述中，不正确的是_____。

A．测试以发现故障或缺陷为目的

B．测试可以找出软件中存在的所有缺陷和错误

C．执行有限测试用例并发现错误

D．检查软件是否满足定义的各种需求

2.【单选】软件测试是为了检查出并改正尽可能多的错误，不断提高软件的_____。

A．功能和效率　　B．设计和技巧　　C．质量和可靠性　　D．质量和效能

3.【单选】导致软件缺陷的最大原因来自_____。

A．软件产品规格说明书　　B．软件设计

C．软件编码　　D．数据输入错误

4.【单选】软件测试的对象包括_____。

A．目标程序和相关文档　　B．源程序、目标程序、数据及相关文档

C．目标程序、操作系统和平台软件　　D．源程序和目标程序

5.【单选】识别测试的任务、定义测试的目标以及为实现测试目标和任务的测试活动规格说明。上述行为主要发生在_____阶段。

A．测试计划和控制　B．测试分析和设计　C．测试实现和执行　D.测试结束活动

6.【单选】某测试团队计划持续在一个被测系统中检测到90%～95%的缺陷比率。虽然测试经理认为无论从测试团队角度还是就行业标准而言这已经是一个标准很高的缺陷检测率。但高层管理者对测试结果失望，认为测试团队仍漏检测了太多缺陷。而用户对此系统的使用满意度相对较好，虽有失效发生但总体的负面影响不大。针对上述情况，作为测试经理可应用以下哪项通用测试原则去向高层管理者解释为什么系统中仍会存在未被检测到的缺陷：_____。

A．缺陷集群性　　B．杀虫剂悖论

C．测试依赖于测试内容　　D．穷尽测试是不可能的

7.【单选】瀑布模型表达了一种系统的、顺序的软件开发方法。以下关于瀑布模型的叙述中，正确的是_____。

A．瀑布模型能够非常快速地开发大规模软件项目

B．只有很大的开发团队才使用瀑布模型

C．瀑布模型已不再适合于现今的软件开发环境

D．瀑布模型适用于软件需求确定，开发过程能够采用线性方式完成的项目

8.【单选】敏捷模型不仅是一个开发过程，而且是一类过程的统称，以下选项中不属于敏捷模型的是_____。

A．极限编程（XP）　　B．IPD 模型

C．迭代式增量开发过程（Scrum）　　D．特征驱动软件开发（FDD）

9.【单选】软件测试工作应该开始于_____。

A．需求分析阶段　　B．概要设计阶段　　C．详细设计阶段　　D．编码之后

10.【单选】在下面的描述中，不能体现前置测试模型要点的是_____。

A．前置测试模型将开发和测试的生命周期整合在一起，标识了项目生命周期从开始到结束之间的关键行为，提出业务需求最好在设计和开发之前就被正确定义

B．前置测试将测试执行和开发结合在一起，并在开发阶段以编码—测试—编码—测试的方式来体现，强调对每一个交付的开发结果都必须通过一定的方式进行测试

C．前置测试模型主张根据业务需求进行测试设计，认为需求分析阶段是进行测试计划和测试设计的最好时机

D．前置测试模型提出验收测试应该独立于技术测试，以保证设计及程序编码能够符

合最终用户的需求

11.【单选】软件质量的定义是_____。

A．软件的功能性、可靠性、易用性、效率、可维护性、可移植性

B．满足规定用户需求的能力

C．最大限度地令用户满意

D．软件特性的总和，以及满足规定和潜在用户需求的能力。

12.【单选】在以下选项中，不属于软件功能性的子特性的是_____。

A．适用性　　B．稳定性　　C．准确性　　D．安全性

13.【单选】软件可移植性应从如下_____方面进行测试。

A．可适应性、易安装性、共享性、易替换性

B．可适应性、易安装性、可伸缩性、易替换性

C．可适应性、易安装性、兼容性、易替换性

D．可适应性、成熟性、兼容性、易替换性

14.【单选】在关于软件质量保证和软件测试的描述中，不正确的是_____。

A．软件质量保证和软件测试是软件质量工程的两个不同层面的工作

B．在软件质量保证的活动中也有一些测试活动

C．软件测试是保证软件质量的一个重要环节

D．软件测试人员就是软件质量保证人员

15.【单选】关于软件测试对软件质量的意义，有以下观点：① 度量与评估软件的质量；② 保证软件质量；③ 改进软件开发过程；④ 发现软件错误。其中，正确的是_____。

A．①、②、③　　B．①、②、④

C．①、③、④　　D．①、②、③、④

16.【单选】软件能力成熟度模型（CMM）将软件能力成熟度自低到高依次划分为 5 级。目前，达到 CMM 第 3 级（已定义级）是许多组织努力的目标，该级的核心是_____。

A．建立基本的项目管理和实践来跟踪项目费用、进度和功能特性

B．使用标准开发过程（或方法论）构建（或集成）系统

C．管理层寻求更主动地应对系统的开发问题

D．连续地监督和改进标准化的系统开发过程

## 二、判断题与填空题

1.【判断】一个成功的测试是发现了至今未发现的错误。_____

2.【判断】测试可以证明程序有错，也可以证明程序没有错误。_____

3.【判断】所有的软件测试都应追溯到用户需求。_____

4.【判断】软件测试贯穿于软件定义和开发的整个过程。_____

5.【判断】软件开发模式与软件测试有密切关系，系统测试计划应该在详细设计阶段产生。

6.【判断】V 模型描述了测试阶段和开发过程期间各阶段的对应关系。_____

7.【判断】软件质量度量包含软件的功能特征和非功能特征。_____

8.【判断】TMM 优化级别主要进行质量控制的测试。_____

9.【填空】将瀑布模式与边写边改模式演进、结合，并加入风险评估的软件开发模式

是_____。

10.【填空】在 RUP 流程中，工作流轴上设计 6 个核心工作流程与 3 个核心支撑工作流程，其中核心工作流程包括业务建模工作流程、_____、分析设计工作流程、实现工作流程、_____工作流程和_____工作流程。

11.【填空】定位探索性测试的软件测试模型是_____。

12.【填空】前置测试模型包括两项测试计划技术：第一项技术是_____，为以后提交的程序测试做好初始化准备，验证需求是否可测试；第二项技术是_____。

13.【填空】系统在特定环境下，在给定的时间内无故障运行的概率称为_____，它是对软件设计、开发以及所预定环境下具有特定能力置信度的一种度量，为衡量软件质量主要参数之一。

14.【填空】ISO/IEC 9126 软件质量模型定义软件包含 6 项质量特性：功能性、可靠性、可用性、_____、_____和可移植性。

15.【填空】软件测试成熟度模型（TMM）由 5 个成熟的测试过程级别构成，分别是初始级、可管理级、_____、_____和优化级。

16.【填空】软件测试生命周期包含在软件生命周期中。从大的方面看，测试生命周期主要横跨两个历程，分为_____的测试历程和_____的测试历程。

17.【填空】如果同样的测试用例被一再重复地执行，则会减少测试的有效性。先前没有发现的缺陷反复使用同样的测试用例也不会被重新发现。这种现象在软件测试中称为_____。

18.【填空】一个故障会被应用程序其他部分的某个或某些故障所掩盖，这种现象称为_____。

19.【填空】_____是为特定目的而设计的一组测试条件、输入测试对象的预期输出或预期行为的数据集合或操作序列，它是执行测试的最小单位。

20.【填空题】通过将一个测试用例的输出结果作为另一个测试用例的输入条件，可将多个测试用例组合成为_____。

## 三、简述题

1．简述软件测试的定义及测试的意义。

2．软件工程或软件测试中如何定义软件缺陷？

3．软件测试所涉及的关键问题有哪些？

4．简述软件测试的 7 项原理。

5．简述软件开发的几种常用模式，分析每种模式软件测试策略的不同。

6．简要分析各种软件测试模型的特点。

7．简要描述软件测试的过程。

8．简述软件质量的概念及质量保障体系。

9．分析给出几项你所知道的软件缺陷或软件故障的实例。

10．V 模型是最具有代表意义的软件测试模型，请简单分析 V 模型的优点和缺点。