※第1章 软件测试概述

严格地说，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。通俗地说，软件工程是实现一个大型程序的一套原则方法，即按工程化的原则和方法组织软件开发工作。软件测试是软件工程的一个重要环节，相当于工程领域中的质量检验部分，是提升软件工程质量的重要手段。

本章介绍软件测试的发展史，详细介绍软件测试的目的、原则以及分类，重点对软件测试工具以及自动化测试技术，为后续课程的学习做必要的准备。

⊙1.1 软件测试的发展历程

软件测试是伴随着软件的产生而产生的。早期的软件开发过程中，软件规模都很小、复杂程度低，软件开发的过程混乱无序、相当随意，测试的含义比较狭隘，开发人员将测试等同于“调试”，目的是纠正软件中已经知道的故障，常常由开发人员自己完成这部分的工作。对测试的投入极少，测试介入也比较晚，常常是等到形成代码，产品已经基本完成时才进行测试。

到了上世纪80年代初期，软件和IT行业进入了大发展，软件趋向大型化、高复杂度，软件的质量也越来越重要。这个时候，一些软件测试的基础理论和实用技术开始形成，并且人们开始为软件开发设计了各种流程和管理方法，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。人们还将“质量”的概念融入其中，软件测试定义发生了改变，测试不单纯是一个发现错误的过程，而且将测试作为软件质量保证(SQA)的主要职能环节，包含软件质量评价的内容。Bill Hetzel在《软件测试完全指南》(Complete Guide of Software Testing)一书中指出：“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。测试是对软件质量的度量。”这个定义至今仍被延用。软件开发人员和测试人员开始坐在一起探讨软件工程和测试问题。

软件测试已有了行业标准(IEEE/ANSI),1983 年 IEEE 提出的软件工程术语中给软件测试下的定义是:“使用人工或自动的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之间的差别。”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。它再也不是一个一次性的,而且只是开发后期的活动,而是与整个开发流程融合成一体。软件测试已成为一个专业,需要运用专门的方法和手段,需要专门人才和专家来承担。

进入上世纪 90 年代,软件行业开始迅猛发展,软件的规模变的非常大,在一些大型软件开发过程中,测试活动需要花费大量的时间和成本,而当时测试的手段几乎完全都是手工测试,测试的效率非常低;并且随着软件复杂度的提高,出现了很多通过手工方式无法完成测试的情况,尽管在一些大型软件的开发过程中,人们尝试编写了一些小程序来辅助测试,但是这还是不能满足大多数软件项目的统一需要。于是,很多测试实践者开始尝试开发商业化的测试工具来支持测试,辅助测试人员完成某一类型或某一领域内的测试工作,而测试工具逐渐盛行起来。人们普遍意识到,工具不仅仅是有用的,而且要对今天的软件系统进行充分的测试,工具是必不可少的。测试工具可以进行部分的测试设计、实现、执行和比较的工作。通过运用测试工具,可以达到提高测试效率的目的。测试工具的发展,大大提高了软件测试的自动化程度,让测试人员从繁琐和重复的测试活动中解脱出来,专心从事有意义的测试设计等活动。采用自动比较技术,还可以自动完成测试用例执行结果的判断,从而避免人工比对存在的疏漏问题。设计良好的自动化测试,在某些情况下可以实现“夜间测试”和“无人测试”。在大多数情况下,软件测试自动化可以减少开支,增加有限时间内可执行的测试,在执行相同数量测试时节约测试时间。而测试工具的选择和推广也越来越受到重视。

在软件测试工具平台方面,商业化的软件测试工具已经很多,如捕获/回放工具、Web 测试工具、性能测试工具、测试管理工具、代码测试工具等等,这些都有严格的版权限制且价格较为昂贵,但由于价格和版权的限制无法自由使用,当然,一些软件测试工具开发商对于某些测试工具提供了 Beta 测试版本以供用户有限次数使用。幸运的是,在开放源码社区中也出现了许多软件测试工具,已得到广泛应用而且已经相当成熟和完善。

⊙1.2 软件测试的目的

简单地说,软件测试就是为了发现错误而执行程序的过程。在 IEEE 提出的软件工

程标准术语中，软件测试被定义为:“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，归根结底，软件测试是为了保证软件质量。软件测试是一个找错的过程。软件测试的过程亦是程序运行的过程。程序运行需要数据，为测试设计的数据称为测试用例。测试用例的设计原则是尽可能暴露程序中的错误。

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。无论是软件从业人员、专家和学者做了多大的努力，软件错误仍然存在。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统统消灭，以期提高软件的质量。一个成功的测试用例在于发现了至今尚未发现的缺陷。

软件测试(software testing)，描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出与预期输出间的审核或者比较过程。软件测试的经典定义是:在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。软件测试的目的是以最少的人力、物力和时间找出软件中潜在的各种错误和缺陷，通过修正各种错误和缺陷提高软件质量，回避软件发布后由于潜在的软件缺陷和错误造成的隐患所带来的商业风险。

软件测试是使用人工操作或者软件自动运行的方式来检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别的过程。Glenford J. Myers 曾对软件测试的目的提出过以下观点:

(1)测试是为了发现程序中的错误而执行程序的过程。

(2)好的测试方案是极可能发现迄今为止尚未发现的错误的测试方案。

(3)成功的测试是发现了至今为止尚未发现的错误的测试。

(4)测试并不仅仅是为了找出错误。通过分析错误产生的原因和错误的发生趋势，可以帮助项目管理者发现当前软件开发过程中的缺陷，以便及时改进。

(5)这种分析也能帮助测试人员设计出有针对性的测试方法，改善测试的效率和有效性。

(6)没有发现错误的测试也是有价值的，完整的测试是评定软件质量的一种方法。

(7)另外，根据测试目的的不同，还有回归测试、压力测试、性能测试等，分别为了检验修改或优化过程是否引发新的问题、软件所能达到处理能力和是否达到预期的处理能力等。

软件测试的目的往往包含如下内容:

(1)发现一些可以通过测试避免的开发风险。

(2)实施测试来降低所发现的风险。

(3)确定测试何时可以结束。

(4)在开发项目的过程中将测试看作是一个标准项目。

⊙1.3 软件测试的原则

在软件测试过程中，应注意以下几个原则：

(1)测试应该尽早进行，最好在需求阶段就开始介入，因为最严重的错误不外乎是系统不能满足用户的需求。

(2)程序员应该避免检查自己的程序，软件测试应该由第三方来负责。

(3)设计测试用例时应考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下要制造极端状态和意外状态，如网络异常中断、电源断电等。

(4)应该充分注意测试中的群集现象。

(5)对错误结果要进行一个确认过程。一般由 A 测试出来的错误，一定要由 B 来确认。严重的错误可以召开评审会议进行讨论和分析，对测试结果要进行严格地确认，是否真的存在这个问题以及严重程度等。

(6)制定严格的测试计划。一定要制定测试计划，并且要有指导性。测试时间安排尽量宽松，不要希望在极短的时间内完成一个高水平的测试。

(7)妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便。

⊙1.4 软件测试的分类

软件测试方法很多，不同的出发点划分出不同的测试方法。

(1)从是否关心软件内部结构和具体实现的角度划分

A. 白盒测试

白盒测试又称结构测试或逻辑驱动测试，与黑盒测试功能正好相反，它是知道产品内部工作过程，检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条路径是否都能按预订要求正确工作。白盒测试的主要方法有逻辑驱动、代码检查、路径测试等。

白盒测试是基于源代码下的测试，需要了解程序的架构、具体需求以及编写程序的技巧，能够检查一些程序规范、指针、变量、数字越界等问题。可以发现以下类型的错误：变量没有声明、无效引用、数字越界、死循环、函数本身没有析构、参数类型不匹配、调用系统的函数没有考虑到系统的兼容性等。

B. 黑盒测试

黑盒测试也称为功能测试或数据驱动测试，它是通过测试来检测已知产品的功能是否能

正常使用。在测试时，把程序看做一个不能打开的黑盒子，在完全不考虑程序内部结构和内部特性的情况下，检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能正确接收输入数据并输出相应正确的输出结果。

黑盒测试可发现以下类型的错误：功能错误、功能遗漏、界面错误、数据结构或外部数据库访问错误、性能错误、初始化和终止错误等。

C. 灰盒测试

灰盒测试，是介于白盒测试与黑盒测试之间的一种测试，灰盒测试多用于集成测试阶段，主要用于测试各个组件之间的逻辑关系是否正确，采用桩驱动，把各个函数按照一定的逻辑串起来，达到在产品还没有界面的情况下输出结果的目的，不仅关注输出、输入的正确性，同时也关注程序内部的情况。灰盒测试不像白盒那样详细、完整，但又比黑盒测试更关注程序的内部逻辑，常常是通过一些表征性的现象、事件、标志来判断内部的运行状态。

(2)从是否执行程序的角度划分

A. 静态测试

静态测试也称为静态分析，是指不运行被测程序，仅通过分析或检查源程序的语法、结构、过程、接口等来确认程序的正确性。对需求规格说明书、软件设计说明书、源程序做结构分析、流程图分析来查找错误。静态方法通过程序静态特性的分析，找出欠缺和可疑之处，例如不匹配的参数、不适当的循环嵌套和分支嵌套、不允许的递归、未使用过的变量、空指针的引用和可疑的计算等。静态测试结果可用于进一步的查错，并为测试用例选取提供指导。

B. 动态测试

动态测试方法是指通过运行被测程序，检查运行结果与预期结果的差异，并分析运行效率、正确性和健壮性等性能。这种方法由三部分组成：构造测试用例、执行程序、分析程序的输出结果。

(3)从软件开发的过程按阶段划分

A. 单元测试

单元测试，是指对软件中的最小可测试单元进行检查和验证。对于单元测试中单元的含义，一般来说，要根据实际情况去判定其具体含义，如 C 语言中单元是指一个函数，Java 里单元是指一个类，图形化的软件中可以指一个窗口或一个菜单等。总的来说，单元就是人为规定的最小的被测功能模块。单元测试是在软件开发过程中要进行的最低级别的测试活动，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

B. 集成测试

集成测试也叫组装测试、联合测试，是单元测试的逻辑扩展。集成测试是在单元测试的基础上，测试在将所有的软件单元按照概要设计规格说明的要求组装成模块、子系统或系统的过程中各部分工作是否达到或实现相应技术指标及要求的活动。它最简单的形式是：把两

个已经测试过的单元组合成一个组件，测试它们之间的接口。

D. 系统测试

系统测试是将经过集成测试的软件，作为计算机系统的一个部分，与系统中其他部分结合起来，在实际运行环境下对计算机系统进行的一系列严格有效的测试，以发现软件潜在的问题，保证系统的正常运行。

E. 验收测试

验收测试是部署软件之前的最后一个测试操作。在软件产品完成了单元测试、集成测试和系统测试之后，产品发布之前所进行的软件测试活动。它是技术测试的最后一个阶段，也称为交付测试。验收测试的目的是确保软件准备就绪，并且可以让最终用户将其用于执行软件的既定功能和任务。

F. 回归测试

回归测试是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。回归测试作为软件生命周期的一个组成部分，在整个软件测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。在渐进和快速迭代开发中，新版本的连续发布使回归测试进行的更加频繁，而在极端编程方法中，更是要求每天都进行若干次回归测试。

G. Alpha 测试(α 测试)

α 测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。α 测试的目的是评价软件产品的 FLURPS(即功能、局域化、可使用性、可靠性、性能和支持)。尤其注重产品的界面和特色。α 测试可以从软件产品编码结束之时开始，或在模块(子系统)测试完成之后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。α 测试即为非正式验收测试。

由于本级的测试过程是可重复、已定义、已管理和已测量的，因此软件组织能够优化调整和持续改进测试过程。测试过程的管理为持续改进产品质量和过程质量提供指导，并提供必要的基础设施。

H. Beta 测试

Beta 测试由软件的最终用户们在一个或多个场所进行。与 Alpha 测试不同，开发者通常不在 Beta 测试的现场，因 Beta 测试是软件在开发者不能控制的环境中的“真实”应用。用户 Beta 测试过程中遇到的一切问题(真实的或想像的)，并且定期把这些问题报告给开发者。接收到在 Beta 测试期间报告的问题之后，开发者对软件产品进行必要的修改，并准备向全体客户发布最终的软件产品。

测试过程按 4 个步骤进行，即单元测试、集成测试、确认测试和系统测试及发布测试。开始是单元测试，集中对用源代码实现的每一个程序单元进行测试，检查各个程序模块是否正

确地实现了规定的功能。集成测试把已测试过的模块组装起来，主要对与设计相关的软件体系结构的构造进行测试。确认测试则是要检查已实现的软件是否满足了需求规格说明中确定了的各种需求，以及软件配置是否完全、正确。系统测试把已经经过确认的软件纳入实际运行环境中，与其他系统成分组合在一起进行测试。

思考题：

1. 软件测试的目的是什么？
2. 按测试技术划分，软件测试分为哪几类？
3. 简单叙述软件测试的原则。

※第 2 章　软件测试的基本知识

本章介绍了软件测试的模型，详细介绍了软件测试的测试用例，重点介绍软件测试流程的几个阶段；并且简单介绍了测试案例。

⊙2.1　软件测试的模型

软件测试模型是软件测试工作的框架，描述了软件测试所包含的主要活动，这些活动之间的相互关系。目前软件测试模型主要有 V 模型、W 模型、H 模型、X 模型和前置模型等。

2.1.1 V 模型

在软件测试方面，V 模型是最广为人知的模型，如图 2－1 所示。V 模型已存在了很长时间，和瀑布开发模型有着一些共同的特性，由此也和瀑布模型一样地受到了批评和质疑。V 模型中的过程从左到右，描述了基本的开发过程和测试行为。V 模型的价值在于它非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程期间各阶段的对应关系。

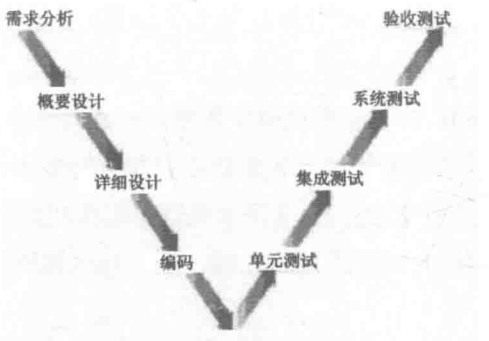

图 2－1　V 模型

V 模型具有如下特点：

(1)V 模型是最具有代表意义的测试模型。

(2)V 模型是软件开发瀑布模型的变种，它反映了测试活动与分析和设计的关系。

从左到右，描述了基本的开发过程和测试行为，非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程期间各阶段的对应关系。

左边依次下降的是开发过程各阶段，与此相对应的是右边依次上升的部分，即各测试过程的各个阶段。

同时 V 模型也存在以下问题：

(1)测试是开发之后的一个阶段。

(2)测试的对象就是程序本身。

实际应用中容易导致需求阶段的错误一直到最后系统测试阶段才被发现。

整个软件产品的过程质量保证完全依赖于开发人员的能力和对工作的责任心，而且上一步的结果必须是充分和正确的，如果任何一个环节出了问题，则必将严重地影响整个工程的质量和预期进度。

2.1.2 W 模型

V 模型的局限性在于没有明确地说明早期的测试，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(如下图)。在模型中不难看出，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。

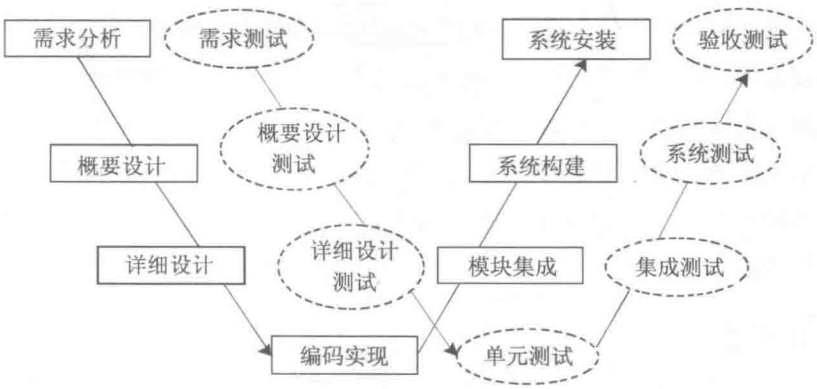

图 2-2 W 模型

W 模型由 Evolutif 公司提出，相对于 V 模型，W 模型更科学。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。如图 2 - 2 所示。

W 模型也有局限性。W 模型和 V 模型都把软件的开发视为需求、设计、编码等一系列串行的活动，无法支持迭代、自发性以及变更调整。

2.1.3 X 模型

X 模型也是对 V 模型的改进，X 模型提出针对单独的程序片段进行相互分离的编码和测试，此后通过频繁的交接，通过集成最终合成为可执行的程序。如图 2 - 3 所示。

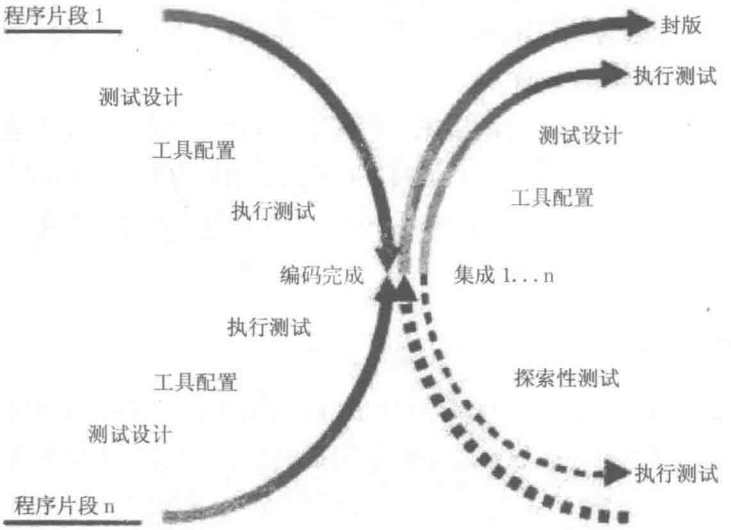

图 2 - 3　X 模型

X 模型的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，通过集成最终成为可执行的程序，然后再对这些可执行程序进行测试。已通过集成测试的成品可以进行封装并提交给用户，也可以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以在各个部分发生。由图中可见，X 模型还定位了探索性测试，这是不进行事先计划的特殊类型的测试，这一方式往往能帮助有经验的测试人员在测试计划之外发现更多的软件错误。但这样可能对测试造成人力、物力和财力的浪费，对测试员的熟练程度要求比较高。

2.1.4 H 模型

H 模型中，软件测试过程活动完全独立，贯穿于整个产品的周期，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。软件测试可以尽早的进行，并且可以根据被测物的不同而分层次进行。如图 2 - 4 所示。

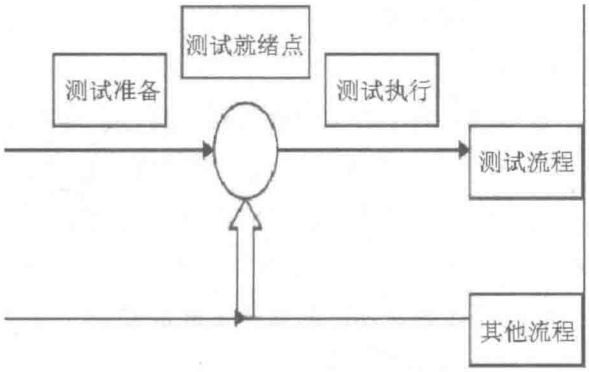

图2-4 H模型

这个示意图演示了在整个生产周期中某个层次上的一次测试“微循环”。图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，只要某个测试达到准备就绪点，测试执行活动就可以开展。

2.1.5 前置模型

前置测试模型则体现了开发与测试的结合，要求对每一个交付内容进行测试。前置测试模型是一个将测试和开发紧密结合的模型，此模型将开发和测试的生命周期整合在一起，随项目开发生命周期从开始到结束每个关键行为。如图2-5所示。

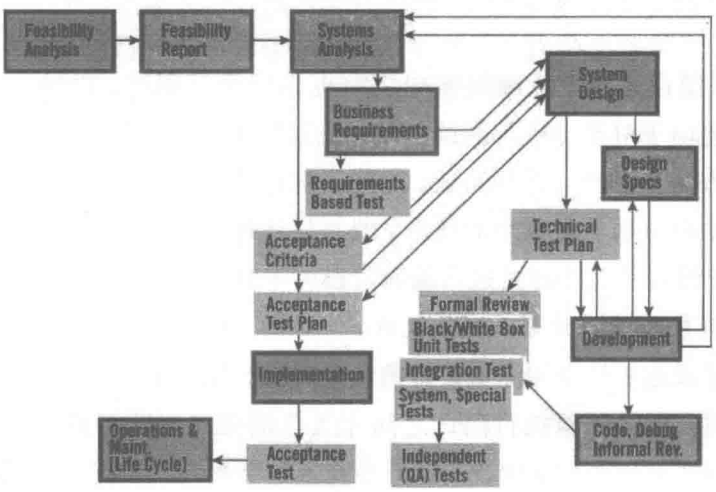

图2-5 前置模型

前置测试模型体现了以下的要点：

（一）开发和测试相结合

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。并且表示了这些行为在项目周期中的价值所在。如果其中有些行为没有得到很好地执行，那么项目成功的可能性就会因此而有所降低。如果有业务需求，则系统开发过程将更有效率。在没有业务需求的情况下进行开发和测试是不可能的。而且，业务需求最好在设计和开发之前就被正确定义。

（二）对每一个交付内容进行测试

每一个交付的开发结果都必须通过一定的方式进行测试。源程序代码并不是唯一需要测试的内容。在图中的绿色框表示了其他一些要测试的对象，包括可行性报告、业务需求说明，以及系统设计文档等。这同 V 模型中开发和测试的对应关系是相一致的，并且在其基础上有所扩展，变得更为明确。

前置测试模型包括 2 项测试计划技术：其中的第一项技术是开发基于需求的测试用例。这并不仅仅是为以后提交上来的程序的测试做好初始化准备，也是为了验证需求是否是可测试的。这些测试可以交由用户来进行验收测试，或者由开发部门做某些技术测试。很多测试团体都认为，需求的可测试性即使不是需求首要的属性，也应是其最基本的属性之一。因此，在必要的时候可以为每一个需求编写测试用例。不过，基于需求的测试最多也只是和需求本身一样重要。一项需求可能本身是错误的，但它仍是可测试的。而且，你无法为一些被忽略的需求来编写测试用例。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来进行验证。验收标准并不仅仅是定义需求，还应在前置测试之前进行定义，这将帮助揭示某些需求是否正确，以及某些需求是否被忽略了。

同样的，系统设计在投入编码实现之前也必须经过测试，以确保其正确性和完整性。很多组织趋向于对设计进行测试，而不是对需求进行测试。Goldsmith 曾提供过 15 项以上的测试方法来对设计进行测试，这些组织也只使用了其中很小的一部分。在对设计进行的测试中有一项非常有用的技术，即制订计划以确定应如何针对提交的系统进行测试，这在处于设计阶段并即将进入编码阶段时十分有用。

（三）在设计阶段进行计划和测试设计

设计阶段是做测试计划和测试设计的最好时机。很多组织要么根本不做测试计划和测试设计，要么在即将开始执行测试之间才飞快地完成测试计划和设计。在这种情况下，测试只是验证了程序的正确性，而不是验证整个系统本该实现的东西。

测试有 2 种主要的类型，这 2 种类型都需要测试计划。在 V 模型中，验收测试最早被定义好，并在最后执行，以验证所交付的系统是否真正符合用户业务的需求。与 V 模

型不同的是，前置测试模型认识到验收测试中所包含的3种成分，其中的2种都与业务需求定义相联系：即定义基于需求的测试，以及定义验收标准。但是，第三种则需要等到系统设计完成，因为验收测试计划是由针对按设计实现的系统来进行的一些明确操作定义所组成，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算成功完成。

技术测试主要是针对开发代码的测试，例如V模型中所定义的动态的单元测试，集成测试和系统测试。另外，前置测试还提示我们应增加静态审查，以及独立的QA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，进行最后的检查。同样的还有特别测试，我们取名特别测试，并把该名称作为很多测试的一个统称，这些测试包括负载测试、安全性测试、可用性测试等，这些测试不是由业务逻辑和应用来驱动的。

对技术测试最基本的要求是验证代码的编写和设计的要求是否相一致。一致的意思是系统确实提供了要求提供的，并且系统并没有提供不要求提供的。技术测试在设计阶段进行计划和设计，并在开发阶段由技术部门来执行。

（四）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－测试的方式来体现。也就是说，程序片段一旦编写完成，就会立即进行测试。通常情况下，先进行的测试是单元测试，因为开发人员认为通过测试来发现错误是最经济的方式。但也可参考X模型，即一个程序片段也需要相关的集成测试，甚至有时还需要一些特殊测试。对于一个特定的程序片段，其测试的顺序可以按照V模型的规定，但其中还会交织一些程序片段的开发，而不是按阶段完全地隔离。

在技术测试计划中必须定义好这样的结合。测试的主体方法和结构应在设计阶段定义完成，并在开发阶段进行补充和升级。这尤其会对基于代码的测试产生影响，这种测试主要包括针对单元的测试和集成测试。不管在哪种情况下，如果在执行测试之前做一点计划和设计，都会提高测试效率，改善测试结果，而且对测试重用也更加有利。

（五）让验收测试和技术测试保持相互独立

验收测试应该独立于技术测试，这样可以提供双重的保险，以保证设计及程序编码能够符合最终用户的需求。验收测试既可以在实施阶段的第一步来执行，也可以在开发阶段的最后一步执行。

前置测试模型提倡验收测试和技术测试沿循两条不同的路线来进行，每条路线分别地验证系统是否能够如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，我们即可确信这是一个正确的系统。

（六）反复交替的开发和测试

在项目中从很多方面可以看到变更的发生，例如需要重新访问前一阶段的内容，或者是跟

踪并纠正以前提交的内容，修复错误，排除多余的成分，以及增加新发现的功能，等等。开发和测试需要一起反复交替地执行。模型并没有明确指出参与的系统部分的大小。这一点和V模型中所提供的内容相似。不同的是，前置测试模型对反复和交替进行了非常明确的描述。

(七)发现内在的价值

前置测试能给需要使用测试技术的开发人员、测试人员、项目经理和用户等带来很多不同于传统方法的内在的价值。与以前的方法中很少划分优先级所不同的是，前置测试用较低的成本来及早发现错误，并且充分强调了测试对确保系统的高质量的重要意义。前置测试代表了整个对测试的新的不同的观念。在整个开发过程中，反复使用了各种测试技术以使开发人员、经理和用户节省其时间，简化其工作。

通常情况下，开发人员会将测试工作视为阻碍其按期完成开发进度的额外的负担。然而，当我们提前定义好该如何对程序进行测试以后，我们会发现开发人员将节省至少20%的时间。虽然开发人员很少意识到他们的时间是如何分配的，也许他们只是感觉到有一大块时间从重新修改中节省下来可用来进行其他的开发。保守地说，在编码之前对设计进行测试可以节省总共将近一半的时间，这可以从以下方面体现出来。

针对设计的测试编写是检验设计的一个非常好的方法，由此可以及时避免因为设计不正确而造成的重复开发及代码修改。通常情况下，这样的测试可以使设计中的逻辑缺陷凸显出来。另一方面，编写测试用例还能揭示设计中比较模糊的地方。总的来说，如果你不能勾画出如何对程序进行测试，那么程序员很可能也很难确定他们所开发的程序怎样才算是正确的。

测试工作先于程序开发而进行，这样可以明显地看到程序应该如何工作，否则，如果要等到程序开发完成后才开始测试，那么测试只是查验开发人员的代码是如何运行的。而提前的测试可以帮助开发人员立刻得到正确的错误定位。

在测试先于编码的情况下，开发人员可以在一完成编码时就立刻进行测试。而且，他会更有效率，在同一时间内能够执行更多的现成的测试，他的思路也不会因为去搜集测试数据而被打断。

即使是最好的程序员，从他们各自的观念出发，也常常会对一些看似非常明确的设计说明产生不同的理解。如果他们能参考到测试的输入数据及输出结果要求，就可以帮助他们及时纠正理解上的误区，使其在一开始就编写出正确的代码。

前置测试定义了如何在编码之前对程序进行测试设计，开发人员一旦体会到其中的价值，就会对其表现出特别的欣赏。前置方法不仅能节省时间，而且可以减少那些令他们十分厌恶的重复工作。

⊙2.2 软件测试的流程

软件测试工作必须要通过制定测试计划、设计测试、实施测试、执行测试、评估测试几个阶段来完成。其流程如图2-6所示。

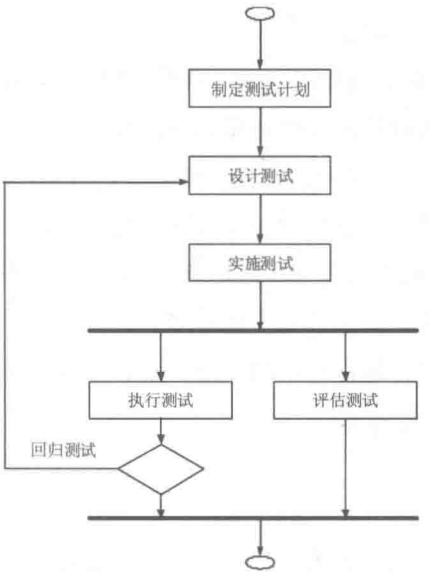

图2-6 软件测试流程

2.2.1 制定测试计划

测试计划是对每个产品，或是对各个开发阶段的产品开展测试的策略。计划的目的是用来识别任务、分析风险、规划资源和确定进度。计划并不是一张时间进度表，而是一个动态的过程，最终以系列文档的形式确定下来。拟定软件测试计划需要测试项目管理人员的积极参与，这是因为主项目计划已经确定了整体项目的一个时间框架，软件测试作为阶段工作必须服从计划和资源上的约定。根据IEEE829-1998软件测试文档编制标准的规定，测试计划包含以下要点：

(1)测试计划标识符，由企业生成的唯一标识符，用于标识测试计划的版本、等级以及关联的软件版本。

(2)介绍，主要介绍被测软件的基本情况和测试范围的概况。

(3)测试项，有功能测试、设计测试、整体测试，可参考需求规格说明、用户指南、操作说明。

(4)需要测试的功能，确定测试需求的范围，需求测试项必须是确定的、可观察、可评测、可量化的。

(5)不需要测试的功能，列出本次测试中不纳入测试的功能。

(6)测试策略，是测试计划的核心内容。测试策略描述测试整体和每个阶段的方法。需要考虑模块、功能、系统、版本、压力、性能等各个因素的影响，尽可能地考虑到细节。主要完成测试技术和工具的确定、测试完成标识的确定。

(7)测试通过/失败的标准的定义。

(8)测试进入和退出的标准的定义。

(9)测试完成需提交的材料，如测试计划、需求分析、测试用例、缺陷报告等。

(10)测试环境，包含测试过程的软件环境和硬件环境。

(11)测试资源，明确测试人力资源和设备资源。

(12)任务安排及人员培训，包含团队成员的详细任务分工，确保任务覆盖全部的测试项。团队成员的业务知识培训，测试工具培训，测试方法培训。

(13)进度安排，明确测试各个环节的完成时间点。

(14)风险预估，指出项目计划过程中的风险，如不现实的完成日期、变更复杂、团队人员流动等，并与项目管理人员交换意见。

2.2.2 测试设计

测试设计阶段要设计测试用例和测试过程，要保证测试用例完全覆盖测试需求。设计测试阶段最重要的是如何将测试需求分解，如何设计测试用例。

1. 如何对测试需求进行分解

对测试需求进行分解需要反复检查并理解各种信息，和用户交流，理解他们的要求。可以按照以下步骤执行。

(1)确定软件提供的主要任务。

(2)对每个任务，确定完成该任务所要进行的工作。

(3)确定从数据库信息引出的计算结果。

(4)对于对时间有要求的交易，确定所要的时间和条件。

(5)确定会产生重大意外的压力测试，包括内存、硬盘空间、高的交易率。

(6)确定应用需要处理的数据量。

(7)确定需要的软件和硬件配置。

(8)确定其他与应用软件没有直接关系的商业交易。

(9)确定安装过程。

(10)确定没有隐含在功能测试中的用户界面要求。

2. 如何设计测试用例

测试用例一般指对一项特定的软件产品进行测试任务的描述，体现测试方案、方法、技术

和策略。值得提出的是，测试数据都是从数量极大的可用测试数据中精心挑选出具有代表性或特殊性的。测试用例是软件测试系统化、工程化的产物，而测试用例的设计一直是软件测试工作的重点和难点。

设计测试用例即设计针对特定功能或组合功能的测试方案，并编写成文档。测试用例应该体现软件工程的思想和原则。

传统的测试用例文档编写有两种方式。一种是填写操作步骤列表:将在软件上进行的操作步骤一步一步详细记录下来，包括所有被操作的项目和相应的值。另一种是填写测试矩阵:将被操作项作为矩阵中的一个字段，而矩阵中的一条条记录，则是这些字段的值。

评价测试用例的好坏有以下两个标准。

(1)是否可以发现尚未发现的软件缺陷?

(2)是否可以覆盖全部的测试需求?

2.2.3 实施测试

实施测试是指准备测试环境、获得测试数据、开发测试规程，以及为该过程挑选和准备辅助测试工具的过程。

1. 准备测试环境

(1)测试技术准备。

(2)配置软件、硬件环境。

(3)人员。

2. 获得测试数据。需要测试的常见情形如下。

(1)正常事务的测试。

(2)使用无效数据的测试。

创建测试数据时主要考虑如下步骤。

(1)识别测试资源。

(2)识别测试情形。

(3)排序测试情形。

(4)确定正确的处理结果。

(5)创建测试事务。

确定实际的测试数据时，必须说明处理测试数据的以下 4 个属性。

(1)深度。

(2)宽度。

(3)范围。

(4)结构。