学习情境 1　熟悉 CVIT 测试过程

知识目标

- 理解软件测试的概念
- 熟悉软件测试的原则
- 掌握软件测试的测试过程
- 掌握软件测试的分类
- 掌握测试用例和 Bug 报告的组成要素
- 熟悉常见的软件测试模型

能力目标

- 能书写测试用例
- 能书写 Bug 报告
- 能熟练掌握软件测试过程角色的职责

引例描述

软件测试是软件工程中保证软件质量的重要环节，越来越受大家的重视。随着当今软件规模和复杂性的日益增加，进行专业化高效软件测试的要求越来越迫切，挑战性极强。大量存在于软件中的各种缺陷需要在软件测试阶段被发现和修正，软件测试员的目标就是找出缺陷，并确保其得以修复，从而保证软件的质量。

本课程以 CVIT 新闻发布系统为测试对象，它是一个基于 B/S 架构的 Web 应用系统，该系统存在诸多问题，例如，该怎样看待测试工作呢，怎样判断软件是否存在 Bug，又如何找出这些 Bug，等等。

任务 1.1　熟悉软件测试的基本概念

任务陈述

目前，软件的规模越来越大，软件的功能越来越多，如何保证软件质量，成为现在 IT 行业最关心的问题，这也是软件测试岗位的工作任务，也是软件测试人员的责任所在。现在对应于新闻发布系统（CVIT），软件测试人员如何去测试，该做哪些测试准备，如何去评价软件缺陷，本任务将带大家了解软件测试的相关概念。

本次任务要求构建 CVIT 系统的框架结构，梳理系统测试的需求，从系统需求说明书中获取测试的功能点、UI 界面测试的要素及性能测试需求；初步制订系统的测试计划，详细罗列测试的要点。

学习目标

- 软件测试的背景
- 软件测试的基本概念
- 软件测试的目的和意义
- 软件测试的分类
- 软件测试的原则

知识准备

1968 年，在北大西洋公约组织的学术会议上科学家第一次提出了“软件工程”这个概念，倡导按照工程化的原则和方法组织软件开发工作。软件工程主要是通过提供规范化的分析设计方法及相应的工具软件，来避免或减少软件错误的发生，为最终杜绝软件危机提供强有力的技术保障。在软件工程中，为了确保质量，软件的含义已不再仅仅是指程序了，而明确为“**软件是程序以及开发、使用和维护程序所需的所有文档**”；“研制软件”也不仅仅是“编程序”了，而明确为“**研制软件过程中所涉及的所有活动，包括分析、设计、编码、测试和维护等**”。

当用工程化的方式有效地管理软件开发的全过程时，程序的编写只是整个工程的一部分，在其前后还有更重要的工作。一般来讲，任何计算机软件都有其生命周期，通常可分为 5 个阶段：需求分析、设计、程序编写、测试、运行和维护。每个阶段都有明确的任务，并生成一定规格的文档交送给下一个阶段，如表 1-1 所示。

表 1-1　软件工程各个阶段的基本情况

阶段		基本任务	工作成果	占开发期的工作量	参与者
开发期	需求分析	理解和表达用户的要求，对开发的软件进行详细的定义	系统需求说明书	20%	用户、系统分析员、高级程序员
	设计	概要设计和详细设计，建立系统的结构，明确系统的实现方式	系统设计说明书、数据说明	15%	系统分析员、高级程序员
	程序编码	写程序	程序	20%	高级程序员、初级程序员
开发期	测试	发现错误和排除错误	可运行的系统	45%其中（模块测试 25% 其他测试 20%）	测试工程师、测试员
运行期		运行和维护	改进的系统		用户、程序员

从表 1-1 可以看出，开发期中各阶段的工作量是不一样的，软件测试占有非常突出的地位，工作量及开销几乎占整个工程的一半，因软件测试是保证软件质量的重要手段，所以越来越引起人们的重视。

1.1.1 软件测试的背景和意义

软件测试在软件生存周期中占有非常突出的位置，是保证软件质量的重要阶段。软件项目的实践证明，为了确保软件产品能够符合用户的需要，必须着眼于整个软件生存周期，在各个阶段进行验证、确认和测试活动，使软件项目在开发完成后，以免发现与用户的需求有较大的差距。

软件在很多领域被广泛使用，然而软件是由程序员编写的，完成后的软件有时并不完美并存在各种各样的缺陷。历史上有很多这样的案例可以证明。

案例 1：迪斯尼的《狮子王》，1994～1995 年

1994 年秋天，迪斯尼公司发布第一张面向儿童的多媒体光盘 Lion King Animated Story Book（狮子王动画故事书）。公司进行了大力宣传，光盘销售额非常可观。但是，在 12 月 26 日，迪斯尼公司的客户服务部却淹没在愤怒的家长和孩子的电话狂潮中。经后来证实，迪斯尼公司因没有对市场投入使用的各种 PC 机型进行测试，光盘软件只能在少数系统中正常工作，但在大众使用的常见 PC 机型的系统中却不能正常工作。

案例 2：美国航天局火星极地登陆，1999 年

1999 年 12 月 3 日，美国航天局的火星极地登陆飞船在试图登陆火星表面时失踪。当错误修正委员会观测到故障后，认定出现失踪的原因极可能是某一个数据位被意外修改。大家一致声讨，问题为什么没有在内部测试时解决。后来发现，登陆飞船经过了多个小组测试，其中一个小组测试飞船的着落点位置，另一个小组测试此后的着陆过程。前一个小组没有注意着地数据位是否置位，后一个小组在开始测试之前重置计算机、清除了数据位，虽两组测试人员都独立很好地完成了工作，但因没沟通未协调而导致出错。

案例 3：爱国者导弹防御系统，1991 年

美国爱国者导弹防御系统首次应用于海湾战争，在对抗伊拉克飞毛腿导弹的防御战争中，几次失利，其中一枚在沙特阿拉伯的多哈击中 28 名美军士兵。经专家分析发现产生此问题的原因是一个软件存在缺陷，由一个很小的时钟错误所积累而导致拖延 14 小时，造成跟踪系统失去精确度。

案例 4：千年虫，1974 年

20 世纪 70 年代的一位程序员，为了节省空间，开发工资系统时将 4 位日期缩减为两位，致使类似系统在 2000 年到来之前，需要更换或升级系统以解决“2000 年错误”的费用超过数亿美元。

案例 5：Intel 奔腾浮点除法，1994 年

在计算机中输入算式（4195835/3145727）*314 5727-4195835，如果结果为零，则计算器没有问题，若不为零，则出现此现象的原因是计算机中使用了老式 Intel 芯片。

以上只是一部分软件运行失败时所发生的历史事件，后果较严重，也可能是灾难性

的。在这些事件中，软件未按照预期需求目标运行，出现了缺陷。随着时间的推移和延长，软件缺陷修复的费用将会呈数十倍地增长，例如，若编写需求说明书时就发现了软件缺陷，花费可能只需几毛钱；若在软件测试时才发现缺陷，花费可能需要几元；若软件缺陷是由客户发现的，花费可能达到几百元。如“迪斯尼狮子王”案例，假如在编写需求说明书时，如果项目成员调查研究过什么机型的 PC 流行，并且明确指出软件需要在该种机型配置上设计和测试，则付出的代价非常小；如果没有这样做，就需要软件测试员去搜集流行 PC 样机并在其上验证，可能会发现软件缺陷，这时付出的代价要高很多。

因此，随着当今软件规模和复杂性的日益增加，进行专业化高效软件测试的要求越来越迫切，且挑战性极强。软件测试员的目标就是找出缺陷，尽可能在面向客户前确保其得以修复，从而保证软件的质量。

那么，到底什么是软件缺陷呢？以下是确认软件缺陷的 5 个规则。

①软件未达到产品说明书中标明的功能。

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当达到的目标。

④软件功能超出了产品说明书中指明的范围。

⑤软件测试人员认为软件难以理解、不易使用，或者最终用户认为该软件使用效果不佳。

例如，计算器的产品说明书明确地声明该产品能准确无误地进行加、减、乘、除运算。假如测试人员，按下加号“+”键，结果计算器什么反应也没有，根据第①条规则，这就是一个软件缺陷，假如相加得到了错误的答案，根据第①条规则，仍然是软件缺陷；产品说明书中声明计算器不会无故崩溃或者停止反应，假如狂敲键盘使得计算器不能接受输入，则根据第②条规则，这也是一个软件缺陷；测试计算器时，电池没电或者电量不足所导致计算不正确，根据第③条规则，也属软件缺陷；假如计算器除了加、减、乘、除，还可以求平方根，这一功能在产品说明书中没有说明，根据第④条规则，还是属软件缺陷；测试人员使用不方便，例如，按键小，布局不好，根据第⑤条规则，这些都是软件缺陷。

1.1.2 软件测试的概念

软件测试在软件开发成本中占有很大的比例，是保证软件质量的主要手段，越来越受到人们的重视。那么，什么是软件测试呢？这一基本概念很长时间以来存在不同的观点。Glen Myers 认为“程序测试是为了发现错误而执行程序的过程”。这一定义明确指出“寻找错误”是测试的目的。相对于“程序测试是证明程序中不存在错误的过程”，Glen Myers 的定义是对的。把证明程序无错当作测试的目的不仅是不正确的，也是完全做不到的，而且对做好测试工作没有任何益处，甚至是十分有害的。从这方面讲，应该接受 Glen Myers 的定义以及其所蕴含的方法论和观点。不过，这个定义规定的范围似乎过于狭窄，使得它受到很大限制。因为如前所述，除去执行程序，还有许多方法去评价和检验一个软件系统。

另外，有些测试专家认为软件测试的范围应当更广泛些。J.B.Goodenough 认为软件测试除了考虑正确性，还应关心程序的效率、稳健性等因素，并且应该为程序调试提供更多的信息。S.T.Redwine 认为，软件测试应该包括几种测试覆盖，分别为功能覆盖、输入域覆盖、输出域覆盖、函数交互覆盖、代码执行覆盖。关于软件测试的范围，A.E.Westley 将软件测试分为 4 个研究方向，即验证技术（目前验证技术仅用于特殊用途的小程序）、静态测试（应逐步从代码的静态测试往高层开发产品的静态测试发展）、测试数据选择、测试技术的自动化。

总的来说，软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码实现的最终审查，是软件质量保证的关键步骤。通常对软件测试的定义如下：

软件测试，是指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**。

事实上，所有发布的软件产品都会因为缺陷而导致用户的困扰和开发者时间、金钱上的额外开支。而这些导致成本风险的软件问题可以通过在软件生命周期的每一个阶段中充分规划、验证和确认而大大降低。广义的软件测试由确认、验证、测试 3 方面组成。

①确认：是指评估将要开发的软件产品是否正确无误、可行和具有价值。其中包含了对用户需求满足程度的评价，确保待开发软件正确无误，是对软件开发构想的检测。

②验证：是指检测软件开发的每个阶段、每个步骤的结果是否正确无误，是否与软件开发各阶段的要求或期望的结果一致。验证意味着确保软件将正确无误地实现软件的需求，开发过程是沿着正确的方向编程。

③测试：与狭隘的测试概念统一，通常要经过单元测试、集成测试、确认测试和系统测试 4 个环节。

在整个软件的生存期，确认、验证、测试分别有其侧重的阶段。确认主要体现在计划阶段、需求分析阶段，也会出现在测试阶段；验证主要体现在设计阶段和编码阶段；测试主要体现在编码阶段。实践中，确认、验证、测试是相辅相成的，确认无疑会产生验证和测试的标准，而验证和测试通常又会帮助完成一些确认工作，特别是在系统测试阶段。因此，软件测试贯穿于软件定义和开发的整个过程。软件开发过程中所产生的需求规格说明、系统设计规格说明及源程序都是软件测试的对象。

1.1.3 软件测试的目的

软件测试的目的决定了如何去组织测试。如果测试的目的是尽可能多地找出错误，那么测试就应该直接针对软件比较复杂的部分或是以前出错比较多的位置。如果测试目的是给最终用户提供具有一定可信度的质量评价，那么测试就应该直接针对在实际应用中会经常用到的商业假设。

不同的机构会有不同的测试目的；相同的机构也可能有不同的测试目的，可能是测试不同区域或是对同一区域的不同层次的测试。在谈到软件测试时，许多人都引用

Grenford J. Myers 的观点：

- 软件测试是为了发现错误而执行程序的过程；
- 测试是为了证明程序有错，而不是证明程序无错误；
- 一个好的测试用例是在于它能发现至今未发现的错误；
- 一个成功的测试是发现了至今未发现的错误的测试。

这种观点可以提醒人们测试要以查找错误为中心，而不是为了演示软件的正确功能。但是仅凭字面意思理解这一观点可能会产生误导，认为发现错误是软件测试的唯一目的，查找不出错误的测试就是没有价值的，事实并非如此。

首先，测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特征，可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时，这种分析也能帮助我们设计出有针对性的检测方法，改善测试的有效性。

其次，没有发现错误的测试也是有价值的，完整的测试是评定测试质量的一种方法。详细而严谨的可靠性增长模型可以证明这一点。例如 Bev Littlewood 发现一个经过测试而正常运行了 *n* 小时的系统有继续正常运行 *n* 小时的概率。

软件测试的目的是保证软件产品的最终质量，在软件开发的过程中，对软件产品进行质量控制。一般来说软件测试应由独立的产品评测中心负责，严格按照软件测试流程，制订测试计划、测试方案、测试规范，实施测试，对测试记录进行分析，并根据回归测试情况撰写测试报告。测试是为了证明程序有错，而不能保证程序没有错误。

1.1.4 软件测试的原则

原则是最重要的，方法应该在这个原则指导下进行。软件测试的基本原则是站在用户的角度，对产品进行全面测试，尽早、尽可能多地发现 Bug，并负责跟踪和分析产品中的问题，对不足之处提出质疑和改进意见。

零缺陷是一种理念，足够好是测试的基本原则。在软件测试过程中，应注意和遵循的具体原则，概括为以下十大项：

（1）所有测试的标准都应建立在用户需求之上。正如我们所知，软件测试的目标就是验证产品的一致性和确认产品是否满足客户的需求，所以测试人员要始终站在用户的角度去看问题、去判断软件缺陷的影响，系统中最严重的错误是那些导致程序无法满足用户需求的缺陷。

（2）软件测试必须基于“质量第一”的思想去开展各项工作，当时间和质量冲突时，时间要服从质量。质量的理念和文化（如零缺陷的“第一次就把事情做对”）同样是软件测试工作的基础。

（3）事先定义好产品的质量标准。有了质量标准，才能依据测试的结果对产品的质量进行正确地分析和评估，例如，进行性能测试前，应定义好产品性能的相关的各种指标。同样，测试用例应确定预期输出结果，如果无法确定测试结果，则无法进行校验。

（4）软件项目一启动，软件测试也就开始了，而不是等程序写完，才开始进行测试。在代码完成之前，测试人员要参与需求分析、系统或程序设计的审查工作，而且要准备测试计划、测试用例、测试脚本和测试环境，测试计划可以在需求模型一完成就开

始准备，详细的测试用例定义可以在设计模型被确定后开始准备。应当把“尽早和不断地测试”作为测试人员的座右铭。

（5）穷举测试是不可能的。甚至一个大小适度的程序，其路径排列的数量也非常大，因此，在测试中不可能运行路径的每一种组合，然而，充分覆盖程序逻辑，并确保程序设计中使用的所有条件是有可能的。

（6）安排第三方进行测试会更客观、更有效。程序员应避免测试自己的程序，为达到最佳的效果，应由第三方来进行测试。测试是带有“挑剔性”的行为，心理状态是测试自己程序的障碍。同时对于需求规格说明的理解产生的错误也很难在程序员本人测试时被发现。

（7）软件测试计划是做好软件测试工作的前提。所以在进行实际测试之前，应制订良好的、切实可行的测试计划并严格执行，特别要确定测试策略和测试目标。

（8）测试用例是设计出来的，不是写出来的，所以要根据测试的目的，采用相应的方法去设计测试用例，从而提高测试的效率，更多地发现错误，提高程序的可靠性。除了检查程序是否做了应该做的事，还要看程序是否做了不该做的事；不仅应选用合理的输入数据，对于非法的输入也要设计测试用例进行测试。

（9）不可将测试用例置之度外，排除随意性。特别是对于做了修改之后的程序进行重新测试时，如不严格执行测试用例，将有可能忽略由修改错误而引起的大量的新错误。所以，回归测试的关联性也应引起充分的注意，有相当一部分最终发现的错误是在早期测试结果中遗漏的。

（10）对发现错误较多的程序段，应进行更深入的测试。一般来说，一段程序中已发现的错误数越多，其中存在的错误概率也就越大。错误集中发生的现象，可能和程序员的编程水平和习惯有很大的关系。

1.1.5 软件测试的分类

软件测试按照不同的划分方法，有不同的分类。按照程序是否执行，可以分为静态测试和动态测试；按照测试用例的设计方法，可以分为白盒测试和黑盒测试；按照开发阶段划分，软件测试可分为单元测试、集成测试、确认测试、系统测试和验收测试；按照测试实施组织划分，软件测试可分为开发方测试、用户测试（β 测试）和第三方测试；按照是否使用工具软件，可以分为手工测试和自动测试。下面简单介绍这些测试。

1. 静态测试和动态测试

原则上讲，可以把软件测试分为两大类，即静态测试和动态测试。静态测试的主要特征是在用计算机测试源程序时，计算机并不真正运行被测试的程序。这说明静态测试一方面要利用计算机作为对被测程序进行特性分析的工具，它与人工测试有着根本的区别；另一方面它并不真正运行被测程序，只进行特性分析，这是和动态测试不同的方面。因此，静态测试常被称为“静态分析”，静态分析是对被测程序进行特性分析的一些方法的总称。

值得注意的是，静态分析并不等同于编译系统，编译系统虽也能发现某些程序错误，但这些错误远非软件中存在的大部分错误，静态分析的查询和分析功能是编译程序所不能代替的。目前，已经开发出一些静态分析系统作为软件测试的工具，已被当作一

种自动化的代码校验方法。不同的方法有各自的目标和步骤，侧重点不同。

静态测试阶段的任务主要表现为以下方面：①检查算法的逻辑正确性；②检查模块接口的正确性；③检查输入参数是否有合法性检查；④检查调用其他模块的接口是否正确；⑤检查是否设置了适当的出错处理；⑥检查表达式、语句是否正确，是否含有二义性；⑦检查常量或全局变量使用是否正确；⑧检查标识符的使用是否规范、一致；⑨检查程序风格的一致性、规范性；⑩检查代码是否可以优化，算法效率是否最高；⑪检查代码注释是否完整，是否正确反映了代码的功能。静态测试包括代码检查、静态结构分析、代码质量度量等，既可由人工进行，也可借助软件工具自动进行。

（1）代码检查。代码检查包括桌面检查、代码审查等，主要检查代码和设计的一致性，代码对标准的遵循、可读性，代码逻辑表达的正确性，代码结构的合理性等方面。代码检查的具体内容有变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等。代码检查的优点主要体现在实际使用中，代码检查比动态测试更有效率，能快速找到缺陷，发现 30%～70%的逻辑设计和编码缺陷。代码检查发现的是问题本身而非征兆。代码检查的缺点是耗费时间长，而且代码检查需要检查人员知识和经验的积累。

（2）静态结构分析。静态结构分析主要是以图形的方式表现程序的内部结构。例如，函数调用关系图、函数内部控制流图。其中，函数调用关系图以直观的图形方式描述一个应用程序中各个函数的调用和被调用关系；函数内部控制流图显示一个函数的逻辑结构，由许多节点组成，一个节点代表一条语句或数条语句，连接节点的叫边，边表示节点间的控制流向。

（3）代码质量度量，主要针对软件的可维护性，目前业界主要存在 3 种度量参数：Line 复杂度、Halstead 复杂度和 McCabe 复杂度。其中 Line 复杂度以代码的行数作为计算的基准。Halstead 复杂度以程序中使用到的运算符与运算元数量作为计数目标（直接测量指标），然后计算出程序容量、工作量等。McCabe 复杂度一般称为圈复杂度，将软件的流程图转化为有向图，然后以图论来衡量软件的质量。

动态测试的主要特征是计算机必须真正运行被测试的程序，通过输入测试用例，对其运行情况进行分析，判断期望结果和实际结果是否一致。动态测试包括以下内容：①功能确认与接口测试；②覆盖率分析；③性能分析；④内存分析。

2. 黑盒测试和白盒测试

黑盒测试和白盒测试的关键是测试用例的设计，对任何工程产品都可用这两种方法对其进行测试。其中，基于产品的功能规划进行测试，检查程序各功能是否实现需求，并检查其中的错误，这种测试称为黑盒测试；而基于产品的内部结构来规划测试，检查内部操作是否按规定执行，各部分是否被充分利用，这种测试称为白盒测试。通常，这两类测试方法是从完全不同的起点出发的，两类方法各有侧重，各有优缺点，构成互补关系，在测试的实践中具有有效和实用性，在规划测试时需要把黑盒测试和白盒测试结合起来运用。通常在进行单元测试时多数采用白盒测试，而在确认测试或系统测试中采用黑盒测试的较多。

3. 单元测试、集成测试、确认测试、系统测试和验收测试

软件测试按测试的不同阶段，可分为单元测试、集成测试、确认测试、系统测试和验收测试。其中，单元测试是针对每个单元的编程模块进行的测试，以确保每个模块能按需求工作为目标。集成测试是对已测试的模块进行组装后进行的测试，目的在于检验与软件需求设计相关的程序结构问题。它是检验所开发的软件能否满足所有功能和性能需求的最后手段。系统测试是检验软件产品能否与系统的其他部分（如硬件、数据库及操作人员）协调工作。验收（用户）测试是检验软件产品质量的最后一道工序，是针对软件的功能和性能能否实现用户的需求而进行的检验，同时软件开发人员也应有一定程度的参与。

（1）单元测试

单元测试主要测试 5 个方面的问题，分别为模块接口、局部数据结构、边界条件、独立的路径和错误处理。其中，模块接口测试是对模块接口进行的测试，检查进出程序单元的数据流是否正确。模块接口测试必须在程序其他测试之前进行。局部数据结构测试主要测试在模块工作过程中，模块内部的数据能否保持完整性，包括内部数据的内容、形式及相互关系不发生错误。

在单元测试中，最主要的测试是针对路径的测试。测试用例必须能够发现由于计算错误、不正确的判定或不正常的控制流而产生的错误。

在单元测试时，如果模块不是独立的程序，需要设置一些辅助测试模块。辅助测试模块有两种：第一种是驱动模块，用来模拟被测模块的上一级模块，相当于被测模块的主程序。能接收数据，将相关数据传送给被测模块，启动被测模块，并打印相应的结果。第二种是桩模块，用来模拟被测模块工作过程中所调用的模块。通常只进行很少的数据处理。驱动模块和桩模块是测试成本中额外的开销，虽然在单元测试中必须编写，但并不需要作为最终的产品提供给用户。被测模块、驱动模块和桩模块共同构成了如图 1-1 所示的单元测试环境。

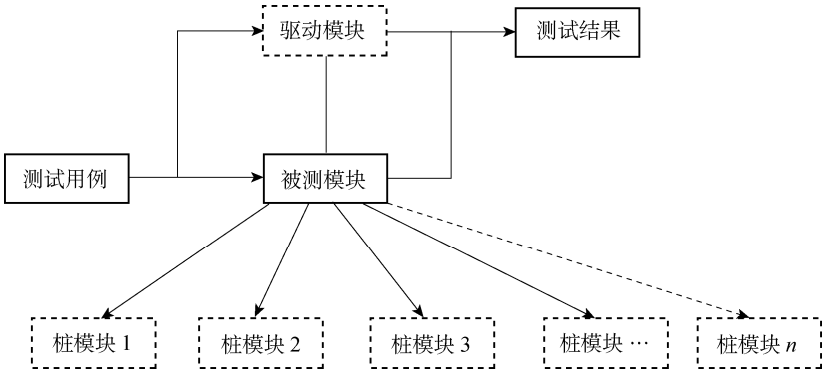

图 1-1　单元测试环境

（2）集成测试

集成测试，也叫组装测试或者联合测试，在单元测试基础上，将所有模块按照设计要求集成为子系统或系统，进行测试。

集成测试通常有两种集成方式，分别为非增量式测试和增量式测试。其中，非增量式测试采用一步到位的方法来构造测试模块。对所有模块进行个别的单元测试后，按照程序结构图将各模块连接起来，把连接后的程序当作一个整体进行测试。这种集成方式的缺点是当一次集成的模块较多时，采用非增量式测试容易出现混乱，因为测试时可能发现了许多故障，这为每一个故障定位和纠正带来非常大的困难，并且在修正一个故障的同时，可能又引入了新的故障，新旧故障混杂，很难判定出错的具体原因和位置。

增量式测试的集成是逐步实现的，即逐次将未曾集成测试的模块和已经集成测试的模块（或子系统）结合成程序包，再将这些模块集成为较大系统，在集成的过程中边连接边测试，以发现连接过程中产生的问题。按照不同的实施次序，增量式集成测试又可以分为三种不同的方法：自顶向下增量式测试、自底向上增量式测试和混合增量式测试。

（3）确认测试

确认测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计要求把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该验证软件的有效性，这就是确认测试的任务，即软件的功能和性能如同用户合理期待的那样。

确认测试又称有效性测试。有效性测试是在模拟的环境下，运用黑盒测试的方法，验证被测软件是否满足需求规格说明书列出的需求。其任务是验证软件的功能和性能及其他特性是否与用户的要求一致。软件的功能和性能要求在软件需求规格说明书中已经明确规定，它包含的信息就是软件确认测试的基础。

确认测试包括安装测试、功能测试、可靠性测试、安全性测试、时间及空间性能测试、易用性测试、可移植性测试、可维护性测试、文档测试。

通过集成测试之后，软件已完成组装，模块接口的错误已被排除，确认测试即可开始。确认测试应检查软件能否按合同需求进行工作，即是否满足软件需求说明书中的确认标准。

目前广泛使用的两种确认测试方式是 α 测试和 β 测试。

①α 测试是指软件开发公司组织内部人员模拟各类用户对即将面市软件产品（称为 α 版本）进行测试，试图发现错误并修正。

它在开发现场进行，开发者在客户使用系统时检查是否存在错误。在该阶段中，需要准备 β 测试的测试计划和测试用例。多数开发者使用 α 测试和 β 测试来识别那些似乎只能由用户发现的错误，其目标是发现严重错误，并确定需要的功能是否被实现。在软件开发周期中，根据功能性特征，所需的 α 测试的次数应在项目计划中规定。

②β 测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用 β 版本，并要求用户报告异常情况、提出批评意见。

它是一种现场测试，一般由多个用户在软件真实运行环境下实施，因此开发人员无法对其进行控制。β 测试的主要目的是评价软件技术内容，发现任何隐藏的错误和边界效应。它还要对软件是否易于使用以及用户文档初稿进行评价，发现错误并进行报告。β 测试也是一种详细测试，需要覆盖产品的所有功能点，因此它依赖于功能性测试。在测试阶段开始前应准备好测试计划，清楚列出测试目标、范围、执行的任务，以及描述测试安排的测试矩阵。客户对异常情况进行报告，并将错误在内部进行文档化以供测试人员和开发人员参考。

（4）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。

系统测试的目的是验证最终软件系统是否满足用户规定的需求，主要内容包括：

①功能性测试，即测试软件系统的功能是否正确，其依据是需求文档，如《产品需求规格说明书》。由于正确性是软件最重要的质量因素，所以功能性测试必不可少。

②健壮性测试，即测试软件系统在异常情况下能否正常运行的能力。健壮性有两层含义，一是容错能力，二是恢复能力。

比较常见的、典型的系统测试包括恢复测试、安全测试、压力测试。下面对这几种测试简单介绍。

①恢复测试：作为一种系统测试，主要关注导致软件运行失败的各种条件，并验证其恢复过程能否正确执行。在特定情况下，系统需具备容错能力。另外，系统失效必须在规定时间段内被更正，否则将导致严重的经济损失。

②安全测试：用来验证系统内部的保护机制，以防止非法侵入。在安全测试中，测试人员扮演试图侵入系统的角色，采用各种办法试图突破防线。因此系统安全设计的准则是要想方设法地使侵入系统所需的代价更加昂贵。

③压力测试：是指在正常资源下使用异常的访问量、频率或数据量来执行系统。在压力测试中可执行以下测试。

- 如果平均中断数量是每秒一到两次，那么设计特殊的测试用例产生每秒 10 次中断。
- 输入数据量增加一个量级，确定输入功能将如何响应。
- 在虚拟操作系统下，产生需要最大内存量或其他资源的测试用例，或产生需要过量磁盘存储的数据。

（5）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。

验收测试是向未来的用户证明系统能够实现预定要求的功能。经集成测试后，按照设计把所有的模块组装成一个完整的软件系统，接口错误也基本被排除，从而应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能如用户所期待的效果呈现。

验收测试，是系统开发生命周期方法论的一个阶段，这时相关的用户和独立测试人员根据测试计划和结果对系统进行测试和接收。让系统用户决定是否接收软件系统，是一项确定软件产品是否能够满足合同或用户所规定需求的软件系统。

实施验收测试的常用策略有三种，分别是：正式验收、非正式验收或 Alpha 测试、Beta 测试。用户选择的策略通常建立在合同需求、组织和公司标准以及应用领域的基础上。

任务实施

1. 详细阅读 CVIT 系统的软件规格需求说明书，按照说明书中提供的软件功能、业务规则、界面要素、性能需求等相关信息，布局软件测试各个测试点。将各个测试点利用工具展现如 XMind 图、Excel 图表等。

2. 认真总结白盒测试方法和黑盒测试方法，初步确定各测试点的测试方法。

3. 整理材料，书写系统的测试方案。

拓展训练

按照如下的测试模板方案，书写 CVIT 的测试方案。

CVIT 系统测试方案

目　录

1. 概述

1.1 编写目的

[说明编写本测试方案的目的]

1.2 读者对象

[本测试方案的合法读者对象为软件开发项目管理者、软件工程师、测试组、系统维护工程师]

1.3 项目背景

[项目简单说明，根据项目的具体情况，方案编写者也可进行详细说明]

2. 测试目的与范围

2.1 测试目的

[说明进行项目测试的目标或所要达到的目的]

2.2 测试参考文档

[参考文档说明]

2.3 测试提交文档

[测试过程需提交文档说明]

2.4 整体功能模块介绍

[在此介绍××系统的功能模块如下表所示]

需求编号	模块名称	功能名称	需求优先级
001	登录		高
002	存放地址	存放地址查看	
003		存放地址搜索	

2.5 相关风险

[风险评估和说明]

3. 测试进度

3.1 测试整体进度安排

测试阶段	时间安排	参与人员	测试工作内容安排	产出
测试方案			● 测试方案	●
测试用例			● 测试用例具体安排	●
第一遍全面测试			●	●
交叉自由测试			●	●

3.2 功能模块划分

模块名称	时间安排	测试负责人	备注
登录			
存放地址			

4. 测试资源

4.1 人力资源分配

角色	人员	主要职责
测试负责人		● 协调项目安排
		●

4.2 测试环境

[描述测试的软件环境（相关软件、操作系统等）和硬件环境]

5. 兼容性测试要求

[描述 B/S 架构的 CVIT 系统在不同浏览器上的兼容性描述]

6. 安全性测试

[描述系统的非法登录等安全性描述]

7. 性能测试

[描述系统的内存占据、响应时间等性能描述]

任务 1.2 软件测试过程

任务陈述

软件工程给我们软件开发提供了一个工程模式，使软件开发的过程遵从模式步骤和阶段要求。同样软件测试是软件开发过程中的一个阶段，此阶段也必须遵照这个模式来进行。当前随着技术的发展，对软件测试技术的研究越来越全面，软件测试过程的运维逐渐完善，软件质量也得以保证。

本任务的目标就是让读者了解软件测试过程，以及测试过程中出现的各项活动安排和任务。以 CVIT 系统为例，阐述软件测试过程的开展和任务要求，以及测试过程管理。

学习目标

- 软件测试的流程
- 软件测试的各阶段任务
- 几种软件测试过程模型
- 软件测试的过程管理

知识准备

软件测试过程是一种抽象的模型，用于定义软件测试的流程和方法。众所周知，开发过程的质量将直接影响测试结果的准确性和有效性。软件测试过程和软件开发过程同样都遵循软件工程原理和管理学原理。

1.2.1 软件测试流程

从一家软件企业的长远发展来看，如果要提高产品的质量首先应当从软件产品流程开始，规范软件产品的开发过程。这是一家软件企业从小作坊的生产方式向集成化规范化的大公司迈进的必经之路，也是从根本上解决质量问题、提高工作效率的一个关键管理规范。

软件产品的开发同其他产品（如汽车）的生产有着共同的特性，即需要按一定的需求规划、设计来进行生产。在工业界，流水线生产方式被证明是一种高效的，且能够较稳定地保证产品质量的一种方式。通过这种方式，不同的工作人员被安排在流程的不同位置，最终为产品的一个目标共同努力，通过流程作业防止人员工作间的内耗，极大地提高工作效率。并且由于其过程来源于成功的实例，因此其最终的产品质量能够满足过程所设定的范围。软件工程在软件的发展过程中吸取了这个经验并应用到软件开发中，这就形成了软件工程过程，也就是开发流程。

不管我们做哪件事情，都有一个循序渐进的过程，从计划到策略再到实现。软件流程就是按照这种思维来定义程序员的开发过程的，根据不同的产品特点和以往的成功经验，定义了从需求到最终产品交付的一整套流程。流程告诉程序员该怎么一步一步地去实现产品，开发过程中可能会有哪些风险，如何避免风险等。由于流程来源于成功的经验，因此，按照流程进行开发可以使得程序员少走弯路，并有效地提高产品质量，提高用户的满意度。

（1）测试工作总体流程图，如图 1-2 所示。

（2）需求、计划、用例阶段流程图，如图 1-3 所示。

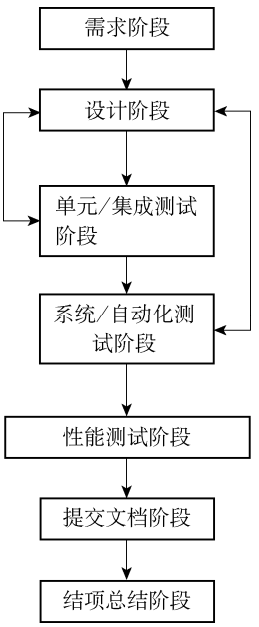

图 1-2 测试工作总流程

说明：集成测试和系统测试的反馈意见可能导致设计文档（需求或数据库）的修改。

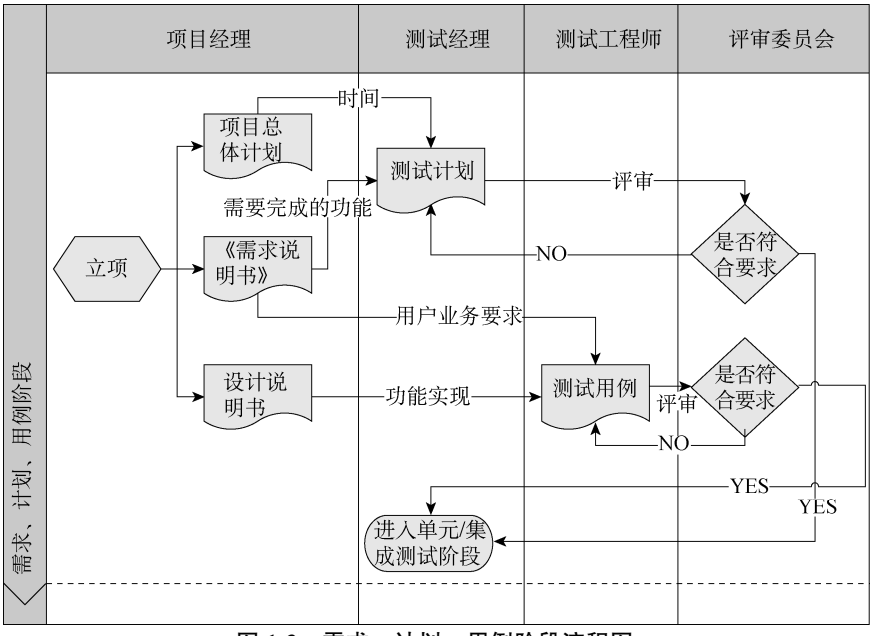

图 1-3　需求、计划、用例阶段流程图

（3）单元/集成测试阶段流程图，如图 1-4 所示。

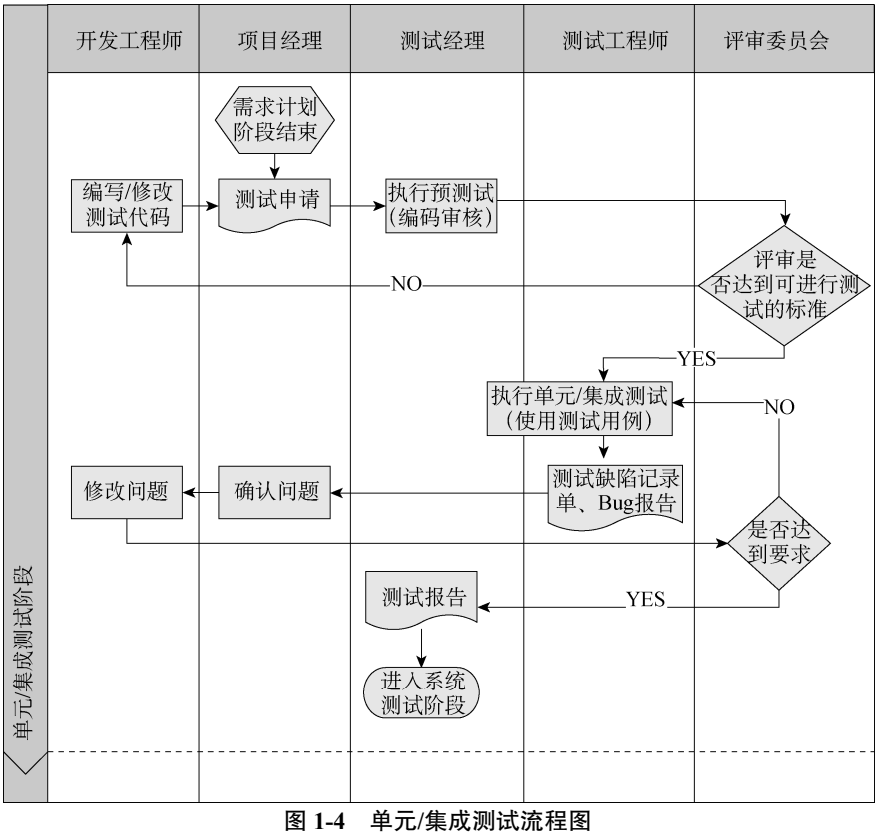

图 1-4　单元/集成测试流程图

（4）系统测试阶段流程图，如图 1-5 所示。

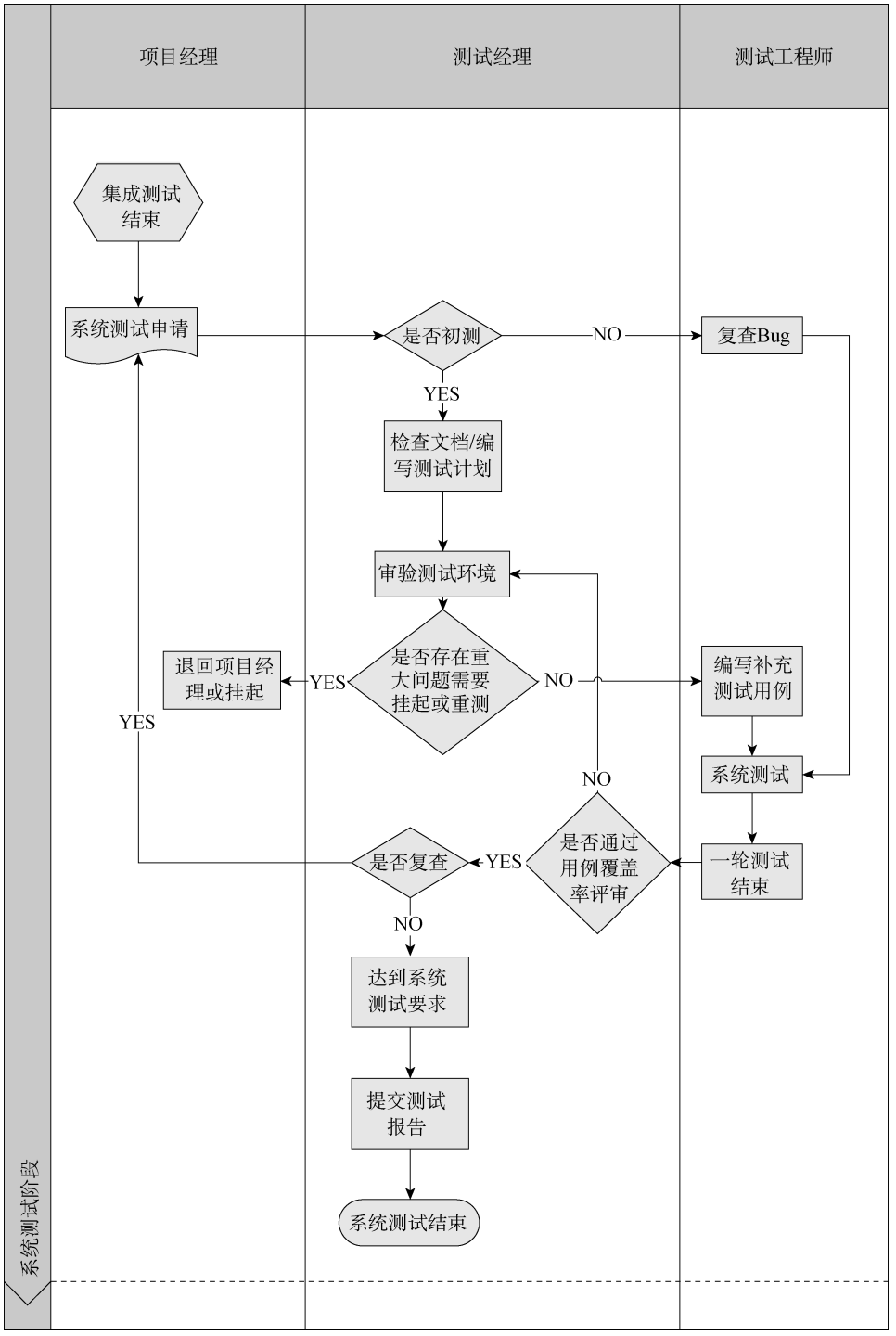

图 1-5　系统测试阶段流程图

（5）验收测试阶段流程图，如图 1-6 所示。

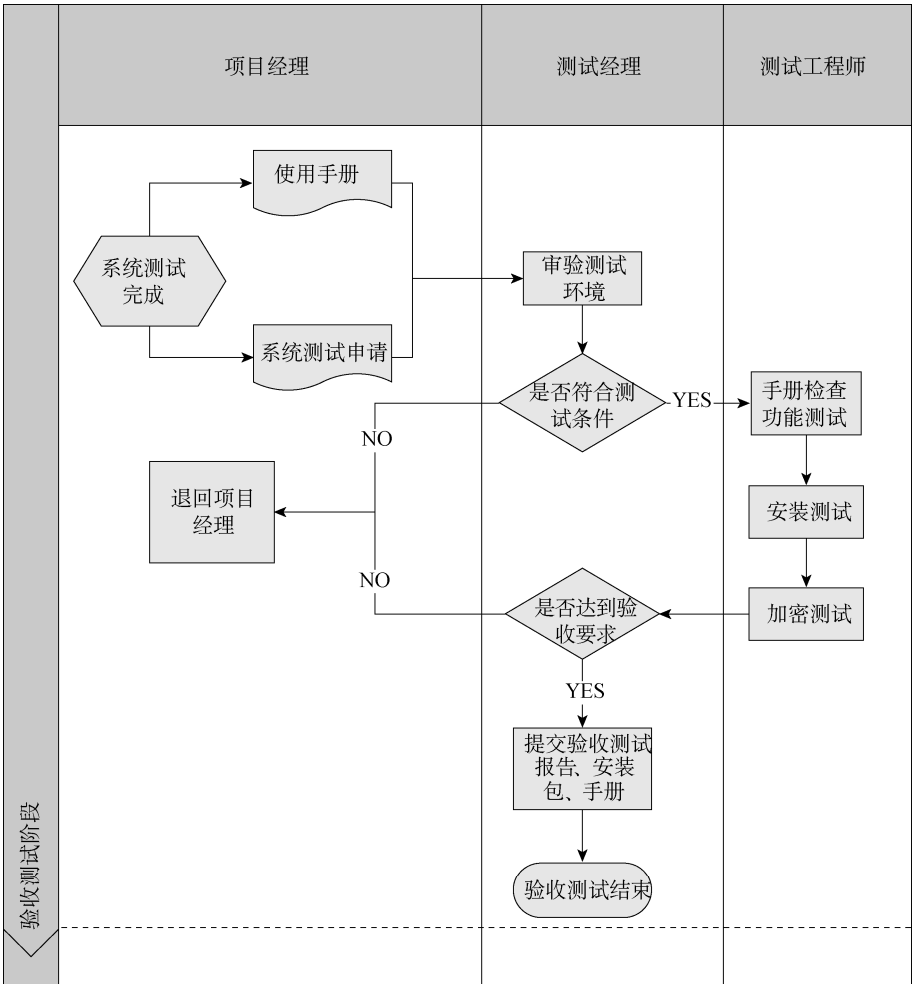

图 1-6　验收测试阶段流程图

说明：验收测试为系统上线前的最后检验，检验方向主要是安装包、安装程序、用户手册、加密设置、基本功能等内容。

1.2.2　测试过程模型

随着软件测试过程管理的发展，软件测试人员通过实践总结了很多测试过程模型，这些模型将测试活动进行了抽象化，并与开发活动进行了有机融合，因此，各模型是测试过程管理的重要参考依据。

1. V 模型

V 模型最早是由 PaulRook 在 20 世纪 80 年代后期提出的，旨在改进软件开发的效率和效果。V 模型反映了测试活动与分析设计活动的关系。如图 1-7 所示，该模型描述了基本的开发过程和测试行为，非常明确地标注了测试过程中存在的不同类型的测试以及这些测试阶段和开发过程期间各阶段的对应关系。

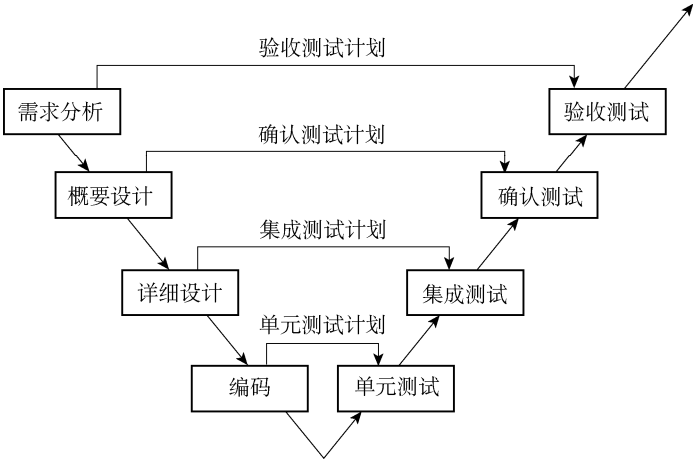

图 1-7　V 模型

V 模型指出，单元和集成测试应检测程序执行是否满足软件设计的要求；系统测试应检测系统的功能、性能的质量特性是否达到系统要求的指标；验收测试应确定软件的实现是否满足用户需求或合同的要求。但 V 模型也存在一定的局限性，其仅仅把测试作为在编码之后的一个阶段，是针对程序寻找错误的活动，而忽视了测试活动对需求分析、系统设计等活动的验证和确认的功能。

2. W 模型

W 模型由 Evolutif 公司提出，相对于 V 模型，W 模型增加了软件各开发阶段应同步进行的验证和确认活动，如图 1-8 所示。W 模型由两个 V 模型组成，分别代表测试与开发过程，明确表示出测试与开发的并行关系。

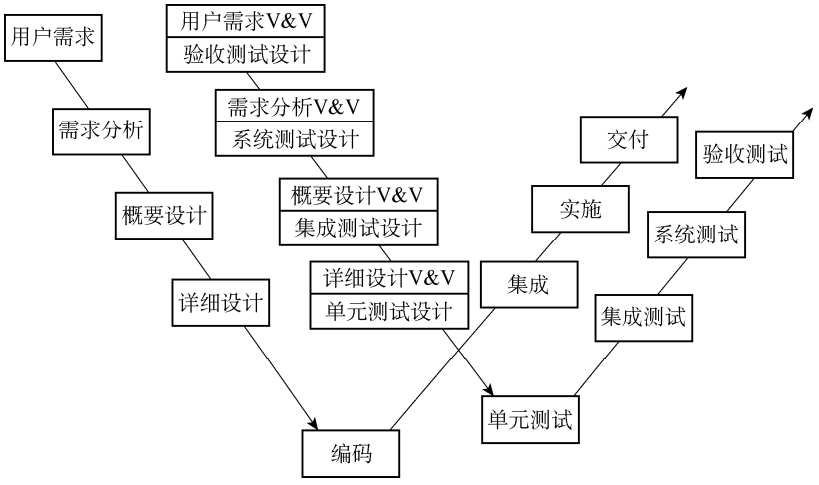

图 1-8　W 模型

W 模型强调测试伴随整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等同样需要测试，因此，测试与开发是同步进行的。W 模型有利于尽早、全面地

发现问题。例如，需求分析完成后，测试人员就应该参与对需求的验证和确认活动，尽早地找出缺陷所在。同时，对需求的测试也有利于及时了解项目难度和测试风险，及早制订应对措施，这将显著减少总体测试时间，并加快项目进度。

但 W 模型也存在局限性。在 W 模型中，需求、设计、编码等活动被视为串行活动，同时，测试和开发活动也保持着一种线性的前后关系，上一阶段结束后才可正式开始下一阶段的工作。这样就无法支持迭代的开发模型。对于当前软件开发复杂多变的情况，W 模型并不能解除测试管理面临的困惑。

3. H 模型

V 模型和 W 模型均存在一些不妥之处。如前所述，两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，而工作实践中，这些活动在大部分时间内都可交叉进行，所以，相应的测试之间也存在严格的次序关系。同时，各层次的测试（单元测试、集成测试、系统测试）也存在反复触发、迭代的关系。

为了解决以上问题，有专家提出了 H 模型。H 模型将测试活动完全独立，形成一个完全独立的测试流程，将测试准备活动和测试执行活动清晰地表现出来，如图 1-9 所示。

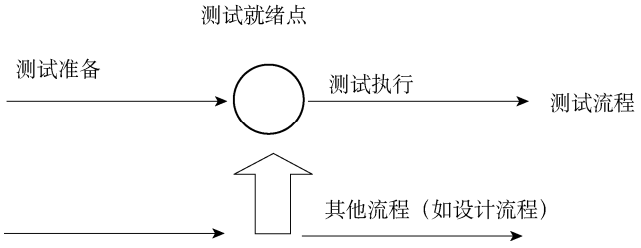

图 1-9　软件测试 H 模型

图 1-9 仅仅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，如设计流程或编码流程。H 模型揭示的原理为：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按照某个次序先后进行，也可能是反复的，只要某个测试达到准备关键点，测试执行活动就可以开展。

4. 其他模型

除上述几种常见模型，业界还流传着其他几种模型，例如前置测试模型、X 模型等。前置测试模型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。X 模型提出针对单独的程序片段进行相互分离编码和测试。此后通过频繁的交接，经集成最终合成为可执行的程序。这些模型都针对其他模型的缺点提出了一些修改意见，但本身也可能存在一些不周全的地方。所以在测试过程中，正确选取过程模型或者根据自身的实际情况选择/制订一个模型是非常关键的问题。

1.2.3　测试过程管理

当今，在软件产品比较发达的国家，软件测试已经成为一个独立的产业，许多软件公司纷纷建立了独立的测试团队。我国的软件测试起步较晚，但随着我国软件产业的蓬