21世纪

高等院校计算机系列教材

软件测试技术

曲朝阳　刘志颖　等编著

中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校计算机系列教材

软件测试技术

曲朝阳　刘志颖　等编著

中国水利水电出版社

内 容 提 要

本书详尽地阐述了软件测试领域中的一些基本理论和实用技术。首先从软件测试的基本原则，以及常用的软件测试技术入手，介绍了与软件测试领域相关的基础知识。然后，分别从单元测试、集成测试和系统测试3个层面深入分析了如何选择和设计有效的测试用例，制定合适的测试策略等主题。最后，讨论了面向对象的软件测试和软件测试自动化技术。附录中还收录了常见的软件错误，供读者参阅。

本书作为软件测试的实际应用参考书，除了力求突出基本知识和基本概念的表述外，更注重软件测试技术的运用，在介绍诸多知识点的过程当中结合直观形象的图表或实际案例进行深入浅出的分析，从而使读者可以更好地理解和掌握软件测试理论知识，并迅速地运用到实际测试工作中去。

本书适合作为各层次高等院校计算机及相关专业的教学用书，也可作为软件测试人员的参考书。

本书所配电子教案可从中国水利水电出版社网站上免费下载，网址是：http://www.waterpub.com.cn/softdown。

图书在版编目(CIP)数据

软件测试技术 / 曲朝阳等编著. —北京：中国水利水电出版社，2006
（21世纪高等院校计算机系列教材）
ISBN 7-5084-3929-5

Ⅰ. 软… Ⅱ. 曲… Ⅲ. 软件—测试—高等学校—教材 Ⅳ.TP311.5

中国版本图书馆CIP数据核字（2006）第078592号

书　　名	**软件测试技术**
作　　者	曲朝阳　刘志颖　等编著
出版　发行	中国水利水电出版社（北京市三里河路6号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787mm×1092mm　16开本　23.75印张　580千字
版　　次	2006年8月第1版　2006年8月第1次印刷
印　　数	0001—4000册
定　　价	34.00元

前　言

随着信息技术的普及，各种各样的软件已经应用到很多领域，软件设计的复杂程度逐渐增加，开发周期不断缩短。而用户对软件的要求却愈来愈高，不再仅仅关注软件产品功能的先进性，还十分重视对产品质量的稳定性和可靠性的考察，这使得软件开发人员和软件测试人员面临着前所未有的挑战。因此，如何保证软件质量将成为软件工程领域深入研究的课题。

毋庸置疑，优化软件开发过程和提高软件测试人员的技术水平，是保证软件质量的最佳途径，这种观念正在被更多的软件行业人士理解、接受和实施。但软件测试在国内仍处于起步阶段，各种软件测试的方法、技术和标准都还在探索阶段。可以说当今中国的软件测试行业正处于“春秋战国”时期，百家争鸣。一方面，这给行业的创新和发展提供了营养丰富的土壤；另一方面，在测试行业一派“欣欣向荣”的气象背后，也隐藏着巨大的危机。软件质量和测试观点“良莠不齐”，“泥沙俱下”。

在这种情况下，很多高校为了培养更多软件行业急需的软件测试人才，都已开设了软件测试课程，但目前适用于高校教学的软件测试技术方面的教材还寥寥无几。因此，为了适应当前教学的需要，编者在软件测试课程实践的基础上，结合当前软件测试技术的最新发展动态编写了本书。

本书作为软件测试的实际应用参考书，除了力求突出基本知识和基本概念的表述外，更加注重软件测试技术的运用，在介绍很多知识点的过程当中都结合直观形象的图表或实际案例进行深入浅出的分析，从而使读者可以更好地理解和掌握软件测试技术理论知识，并迅速地运用到实际测试工作中去。

本书参考教学时数为 32～40 学时。全书共分 8 章：第 1 章讨论了软件测试的发展历史、软件测试的定义和基本原则；第 2 章介绍了软件测试的基础知识；第 3～5 章结合本书所提供的案例分别介绍了与单元测试、集成测试和系统测试相关的知识以及具体的实施过程；第 6 章介绍了面向对象的软件测试，讨论了面向对象软件测试和传统软件测试的区别，介绍了面向对象软件测试的层次、测试用例和测试驱动程序的设计方法，以及面向对象软件测试的工具 Junit；第 7 章介绍了软件测试自动化，讨论了自动化测试的时机，自动化测试成本的衡量，自动化测试工具的选择和使用；第 8 章介绍了关于软件 BUG 及其管理方面的知识，讨论了软件 BUG 的分类、提交和管理相关的知识。

本书由曲朝阳教授和刘志颖副教授共同编写，李楠、于海洋与杨杰明等老师完成了本书的校对工作，李佳、王敬东与张金伟等研究生为本书的实例做了大量的调试工作，在此一并致谢。

本书在编写过程中，参阅了很多国内外同行的著作或文章，汲取了该领域最新的研究成果。在此，对这些成果的作者表示深深的感谢！

由于编者水平有限，书中难免存在一些错误和不妥之处，希望广大读者批评指正。

编者

2006 年 7 月

目　　录

第 1 章　概述

本章要点

软件测试的发展历史；软件测试技术的分类方法；软件测试原则；软件测试的定义；软件测试同软件开发之间的关系；软件测试与开发模型；软件测试工作流程。

本章目标

- 了解软件测试的发展历程和行业现状
- 掌握软件测试技术的分类
- 理解软件测试的目的和软件测试原则，了解人们对软件测试行业的错误认识
- 掌握软件测试中的基本定义、基本知识
- 理解软件开发与软件测试的关系

1.1　软件测试的发展历程及现状

1.1.1　软件测试的发展历程

随着计算机的诞生——在软件行业发展初期就已经开始实施软件测试，但这一阶段还没有系统意义上的软件测试，更多的是一种类似调试的测试。测试是没有计划和方法的，测试用例的设计和选取也都是根据测试人员的经验随机进行的，大多数测试的目的是为了证明系统可以正常运行。

20 世纪 50 年代后期到 20 世纪 60 年代，各种高级语言相继诞生，测试的重点也逐步转入到使用高级语言编写的软件系统中来，但程序的复杂性远远超过了以前。尽管如此，由于受到硬件的制约，在计算机系统中，软件仍然处于次要位置。软件正确性的把握仍然主要依赖于编程人员的技术水平。因此，这一时期测试理论和方法的发展比较缓慢。

20 世纪 70 年代以后，随着计算机处理速度的提高，存储器容量的快速增加，软件在整个计算机系统中的地位变得越来越重要。随着软件开发技术的成熟和完善，软件的规模也越来越大，复杂度也大大增加。因此，软件的可靠性面临着前所未有的危机，给软件测试工作带来了更大的挑战，很多测试理论和测试方法应运而生，逐渐形成了一套完整的体系，培养和造就了一批批出色的测试人才。

如今在软件产业化发展的大趋势下，人们对软件质量、成本和进度的要求也越来越高，质量的控制已经不仅仅是传统意义上的软件测试。传统的软件测试大多是基于代码运行的，并且常常是在软件开发的后期才开始进行，但大量研究结果表明，设计活动引入的错误占软件开发过程中出现的所有错误数量的 50%~65%。因此，越来越多的声音呼吁，要求有一个规范的软件开发过程。而在整个软件开发过程中，测试已经不再只是基于程序代码进行的活动，而是一个基于整个软件生命周期的质量控制活动，贯穿于软件开发的各个阶段。

1.1.2 软件测试的现状

在我国，软件测试可能还算不上一个真正的产业，软件开发企业对软件测试认识淡薄，软件测试人员与软件开发人员往往比例失调，而在发达国家和地区软件测试已经成了一个产业，微软的开发工程师与测试工程师的比例是1：2，国内一般公司是6：1。很多人认为导致这种现状产生的原因与我们接受的传统教育和开发习惯有相当大的关系。软件行业相对于其他一些行业来说是相当年轻的，开发工作包含了需求管理、分析、设计、测试和部署等工作，由于软件业的历史年轻，而且一般人认为，开发周期前面的工作没有完善之前，比较难于考虑到后面的工作。因此，我们可以看到软件工作大部分的精力都投入在了需求管理、分析、设计3个阶段的开发，造成了这些方面方法论的快速发展，而忽视了测试工作。

总之，与一些发达国家相比，国内测试工作还存在一定的差距。主要体现在测试意识以及测试理论的研究、大型测试工具软件的开发以及从业人员数量等方面。其实，这与中国整体软件的发展水平是一致的，因为我国整体的软件产业水平和软件发达国家水平相比有较大的差距，而作为软件产业重要一环的软件测试，必然也存在着不小的差距。但是，我们在软件测试实现方面并不比国外差，国际上优秀的测试工具，我们基本都有，这些工具所体现的思想我们也有深刻的理解，很多大型系统在国内都得到了很好的测试。

1.2 什么是软件测试

正如食品生产厂家在把产品销售给商家之前要进行合格检验一样，软件企业在把软件提交给客户之前也需要进行严格的测试。如果把所开发出来的软件看作一个企业生产出的产品，那么软件测试就相当于该企业的质量检测部分。简单地说，我们在编写完一段代码之后，检查其是否如我们所预期的那样运行，这个活动就可以看作是一种软件测试工作。

1.2.1 软件测试的定义

从前面的章节中，我们已经了解到软件测试的研究可以追溯到20世纪60年代，至今已有40多年的发展历史，但对于什么是软件测试（software testing），还一直未能达成共识。根据侧重点的不同，主要有以下3种观点：

- IEEE在1983年将软件测试定义为“使用人工或自动手段运行或测定某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别”，该定义明确地提出了软件测试以检验是否满足需求为目标。
- Myers则认为软件测试“是为了发现错误而执行程序的过程”，明确提出了“寻找错误”是测试目的。
- 从软件质量保证的角度看，软件测试是一种重要的软件质量保证活动，其动机是通过一些经济、高效的方法，捕捉软件中的错误，从而达到保证软件内在质量的目的。

上述3种观点实际上是从不同的角度理解测试，是将测试置于不同的环境下得出的结论。事实上，在公开出版的刊物中，有多种不同的关于软件测试的定义，根据这些定义可以认为软件测试是一个在可控的环境中执行软件的过程，目的就是为了验证软件是否按照预期运行。测试过程中的活动既包括“分析”软件，也包括“运行”软件。常常把与分析软件开发中的各种

产品相关的测试活动称为静态测试（static testing）。静态测试包括代码审查、走查和桌面检查。相比之下，把与运行软件有关的测试活动叫做动态测试（dynamic testing）。因此，不能简单地认为软件测试就是程序测试，只有在程序编码结束后才能够进行；而是一项贯穿于整个软件开发过程的工作。测试对象既包括源程序，也包括需求规格说明、概要设计说明、详细设计说明。因此，也有人认为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的最终复审，是软件质量保证的关键步骤。测试包括寻找缺陷，但不包括跟踪漏洞及其修复。测试的重要性在于，它必须保证所开发的软件达到设计时的需求，免除由于软件自身的“缺陷”带来的“漏洞”，最大限度地降低软件开发的成本。

软件测试有两个基本职责，即验证和确认。Schulmeyer 和 Mackenziee（2000）对验证和确认所做的定义是：

验证（verification）：保证开发过程中某一具体阶段的产品与该阶段和前一阶段的需求一致。

确认（validation）：保证最终得到的产品满足系统需求。

有时，初学者常常会混淆软件测试和调试。其实二者是不同的，表现在以下几方面：①调试是一个分析和定位软件 BUG 的过程。可以认为它是一种支持测试，但不能完全代替测试活动。②调试的目的是为了使软件能够正确运行，而测试的目的是为了发现软件中存在的错误。③调试的对象主要是源代码，而测试的对象则是软件开发过程中各个阶段所产生的所有产品。

1.2.2　软件测试生命周期

图 1-1 给出了软件测试生命周期（software testing life cycle）的模型。把测试的生命周期分为几个阶段。前 3 个阶段是引入程序错误阶段，也就是开发过程中的需求规格说明、设计、编码阶段，此时极易引入错误或者导致开发过程中其他阶段产生错误。然后是通过测试发现错误的阶段，这需要通过使用一些适当的测试技术和方法来共同完成。后 3 个阶段是清除程序错误的阶段。其主要任务是进行缺陷分类、缺陷隔离和解决缺陷。其中在修复旧缺陷的时候很可能引进新的错误，导致原来能够正确执行的程序出现新的缺陷。

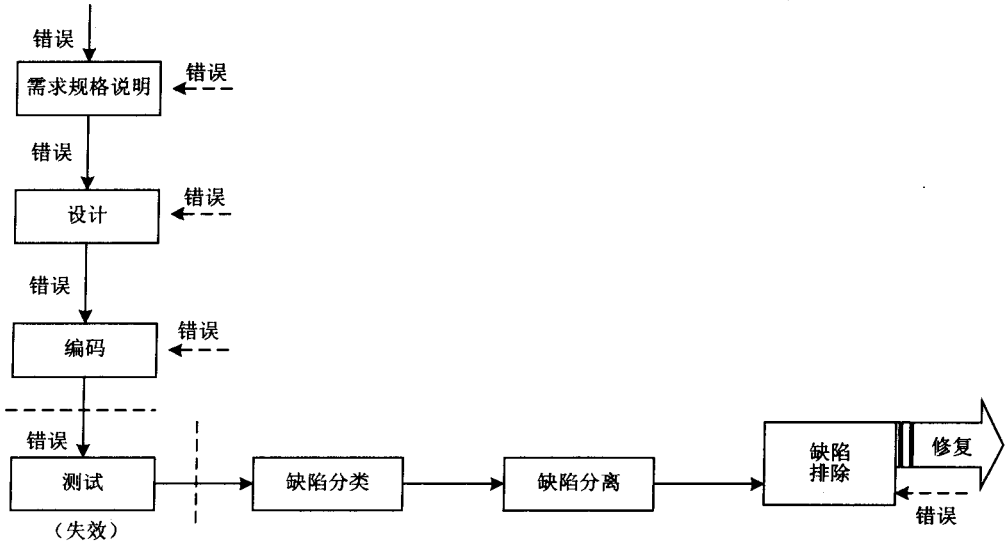

图 1-1　软件测试生命周期

在软件测试生命周期的每个阶段都要完成一些确定的任务，在执行每个阶段的任务时，可以采用行之有效的结构分析设计技术和适当的辅助工具；在结束每个阶段的任务时都进行严格的技术审查和管理复审。最后提交最终软件配置的一个或几个成分（文档或程序）。

1.2.3 软件开发与测试模型

通常情况下，测试过程包括确定要测试什么（测试范围和条件）以及产品如何被测试（制作测试用例），建立测试环境，执行测试，最后再评估测试结果，检查是否达到已完成测试的标准，并报告进展情况等活动。由此可以看出，软件测试不仅仅是执行测试，而是一个包含很多复杂活动的过程，并且这些过程应该贯穿于整个软件开发过程。如果你把测试设计放在最后阶段，就会错过发现构架设计和业务逻辑设计中存在的严重问题的时机，到时候修复这些缺陷将很不方便，因为缺陷已经扩散到系统中去了，所以这样的错误将很难寻找和修复，并且代价更高。读者可能会问，那么如何协调软件测试与开发活动之间的关系？在软件开发的过程中，应该什么时候进行测试？如何更好地把软件开发和测试活动集成到一起？其实这也是软件测试工作人员必须要考虑的几个问题，因为只有这样才能提高软件测试工作的效率，提高软件产品的质量，最大限度地降低软件开发与测试的成本，减少重复劳动。下面我们将介绍几种典型的软件开发与测试模型，这些模型在不同程度上回答了前面所提出的问题。

1. 软件开发与测试 V 模型

V 模型如图 1-2 所示。事实上，在传统开发过程中，仅仅把测试过程作为在需求分析、概要设计、详细设计及编码之后的一个阶段。如：在瀑布模型中，认为测试只是在很多重要开发活动完成后的收尾工作，而不是主要的过程。V 模型对此进行了改进，不再把测试看作是一个事后弥补行为，而是一个同开发过程同样重要的过程。该模型最早由已故的 Paul Rook 在 20 世纪 80 年代后期提出，在英国国家计算中心文献中发布，在欧洲尤其是英国被普遍接受，并把它当作瀑布模型的替代品。

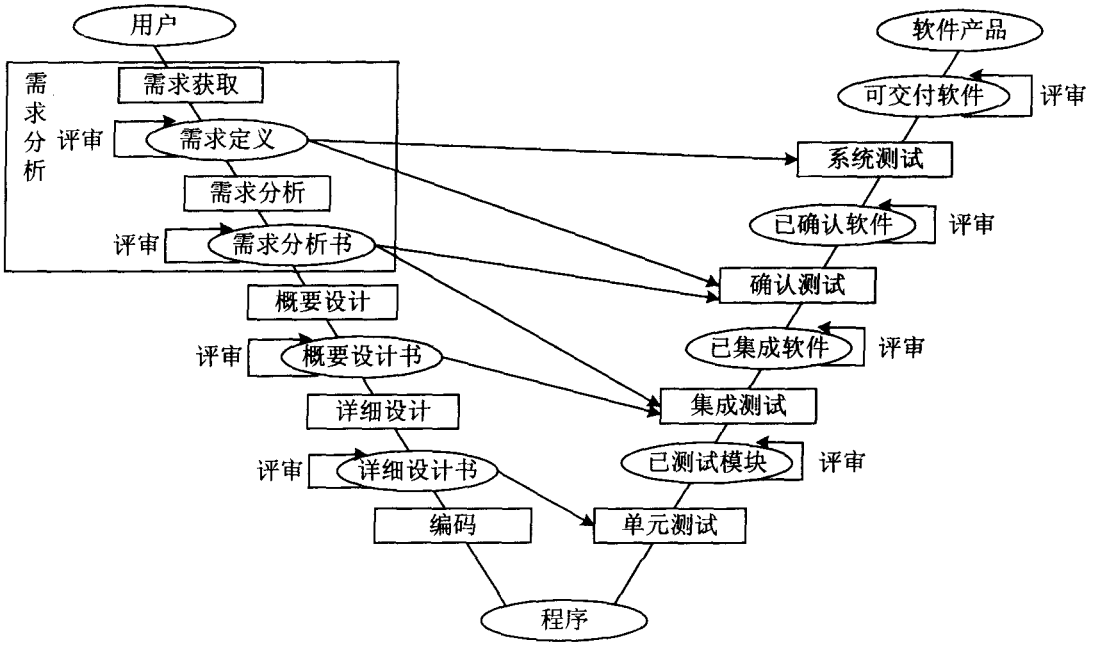

图 1-2 V 模型示意图

如图 1-2 所示的 V 模型描述了一些不同的测试级别，并说明了这些级别所对应的生命周期中不同的阶段。其中，左边下降的部分是开发过程各阶段，与此相对应的右边上升的部分是测试过程的各个阶段。读者要注意，在不同的组织中对测试阶段的命名可能有所不同。在模型图中的开发阶段一侧，先从定义业务需求开始，然后要把这些需求不断地转换到概要设计和详细设计中去，最后开发程序代码。在测试执行阶段一侧，执行先从单元测试开始，然后是集成测试、确认测试和系统测试。

V 模型的价值主要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的对应关系：

- 单元测试的主要目的是根据详细设计说明书来验证和确认每个单元模块是否符合预期的要求，发现编码过程中可能存在的各种错误。
- 集成测试的主要目的是根据概要设计来验证和确认各个模块是否已正确集成到一起，主要是检查各单元与其他模块之间的接口上可能存在的错误。
- 确认测试的主要目的是根据需求分析来验证和确认软件是否符合用户的预期要求。
- 系统测试的主要目的是根据需求定义、验证和确认系统作为一个整体是否能够正常有效地运行，例如：判断系统是否达到了用户预期的性能。

在不同的开发阶段，所引入的缺陷和错误类型不同，因此需要使用不同的测试技术和方法来发现这些缺陷。我们在后面的章节中将对此作具体的介绍。

根据 V 模型的要求，一旦有文档提供，就要及时确定测试条件，编写测试用例，这些工作对每个级别的测试都有意义。当需求提交之后，就需要针对这些需求确定更高级别的测试用例；当概要设计编写完成后，就需要确定测试条件来查找该阶段的设计缺陷。因此如果能尽早提交测试文档，就可以有更多的检查和审阅时间，使测试者可以在项目中能尽早发现规格说明书中存在的问题。这些都说明测试的目的不仅仅是为了评定软件，更重要的是能够尽可能早地找出缺陷所在，从而达到改进项目质量的目的。参与前期工作的测试者可以预先估计可能存在的问题和测试执行的难度，大大减少总体测试时间，提高项目进度。

V 模型常被错误地认为要求开发和测试保持一种线性的前后关系，需要有严格的指令表示上一阶段完全结束，才可正式开始下一个阶段，这样就无法支持迭代、自发性以及变更调整，情况其实并不是这样的。各种模型只是简单地提醒我们，有必要定义一些必须要做什么（需求）——What，然后描述如何做（设计）——How，最后花很多力气来实现（编码）——Do。V 模型所做的是强调每一个开发级别都有一个与之相关联的测试级别，并且建议测试应该在各级别之前进行设计。各模型并没有规定工作量大小，有经验的开发人员通常能够将项目分解为可操作的小阶段，例如在迭代式开发中整个项目被分解为很多小片断，并且忽略各片段的实际大小。此时，模型的 what-how-do 顺序对于按时交付就具有重要意义，而且对于保证每一个阶段目标的实现也非常重要。

V 模型适用于所有类型的开发过程，但并不一定适用于开发和测试过程的所有方面。不管是 GUI 还是批处理、大型机还是 Web、Java 还是 Cobol，都需要单元测试、集成测试、系统测试和验收测试。但是，V 模型本身并不会告诉你如何定义单元测试或集成测试的内容，如何才能使测试工作顺利进行，如何进行具体的测试设计，以及该输入什么样的数据，输出的结果是什么样才正确。

还有一些测试者认为：“是否使用 V 模型要根据项目本身来定。有些项目需要应用 V 模型，

而有些项目不需要。那么，可以只在需要 V 模型的项目中采用。”在实际工作中，这样的做法是不对的，我们应该尽可能地去应用模型中对项目有实用价值的方面，但不要为了使用模型而使用模型，否则便失去了使用模型的实际意义。

正因为 V 模型还存在一些不够完善的地方，随着软件测试技术的不断发展，由 V 模型演化出了很多种软件开发与测试模型，下面介绍的几种测试模型都不同程度地改进了 V 模型的不足之处。

2. 软件开发与测试 W 模型

由于原始问题的复杂性、软件的复杂性和抽象性、软件开发各个阶段工作的多样性以及各种层次人员之间工作的配合关系等因素，使得开发的每一个环节都可能产生错误。如果坚持对各个阶段进行技术评审，就能够尽早发现和预防错误。如图 1-3 所示为软件开发与测试的 W 模型，它形象地说明了软件测试与开发的这种同步性。

与 V 模型相比，在 W 模型中我们很容易就能够看出测试伴随着整个软件开发周期，测试的对象不仅仅是程序还包括需求和设计。应用该模型的优点在于，每个软件开发活动结束后就可以执行相应的测试，如：在需求分析结束后，就可以进行需求分析测试。

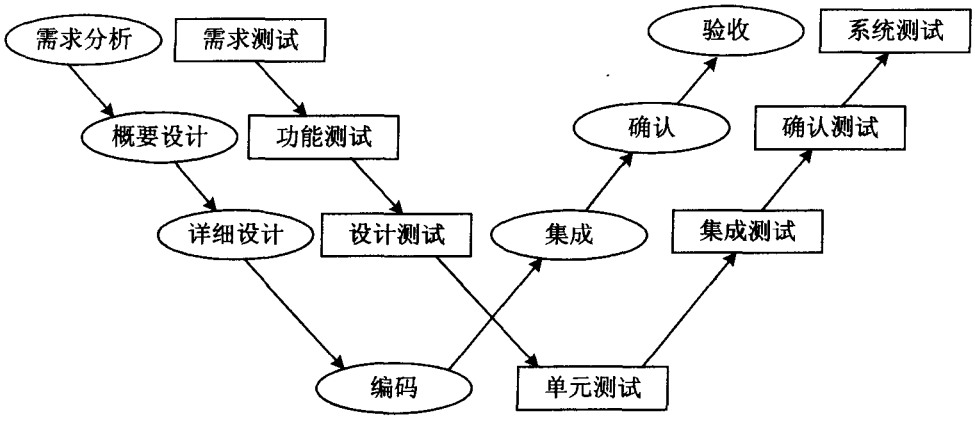

图 1-3 W 模型示意图

3. 软件开发与测试 H 模型

与前两种模型相比，H 模型充分地体现了测试过程，演示了在整个生产周期中，某个（测试）层次上的一次测试“微循环”（可以看作是一个流程在时间上的最小构成单位）。图 1-4 中的“其他流程”可以是任意开发流程，例如设计流程和编码流程，也可以是其他非开发的流程，例如 SQA 流程，甚至是测试流程自身。向上的双线箭头表示在某个时间点，由于“其他流程”的进展（由于先后关系）而引发或者（由于因果关系）触发了测试就绪点，这个时候，只要测试准备活动完成，测试执行活动就可以进行了。

如图 1-4 所示的 H 模型揭示了：

- 软件测试不仅仅指测试的执行，还包括很多其他的活动。
- 软件测试是一个独立的流程，贯穿产品的整个开发周期，与其他流程并发进行。
- 软件测试要尽早准备，尽早执行。
- 软件测试根据被测物的不同是分层次的，不同层次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。

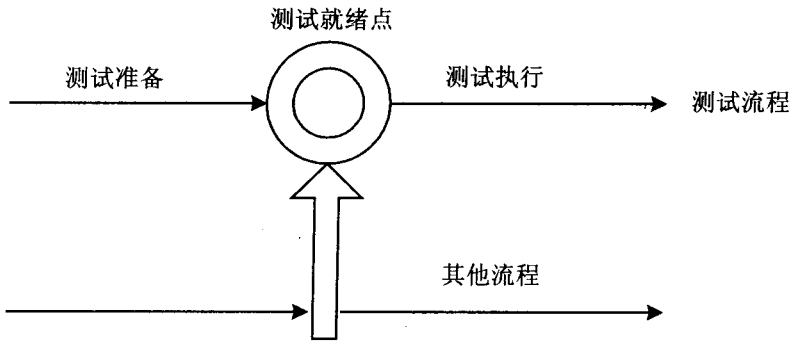

图 1-4　H 模型示意图

1.2.4　与软件测试相关的术语

在软件测试发展的过程中，还有一些大量相关的测试术语。很多读者，尤其是初学者在阅读软件测试相关文献时，常常会因含混不清的测试术语而难以理清头绪。为了使读者能够更好地理解软件测试，本书采用电子电气工程师协会所制定的标准，对一些常见的测试术语进行简单的介绍，读者如想了解更多的软件测试术语可以参考书后的附录。

1. 错误（Error）

程序员在编写代码时会出错，把这种错误称之为 BUG。随着开发过程的进行，错误会不断地放大。例如，需求错误在设计期间会放大，在编写代码时还会进一步放大。

2. 缺陷（Default）

缺陷是错误的结果，更精确地说是错误的表现。而表现的方式有很多种，例如：叙述性文字、数据流框图、层次结构图、源代码等。缺陷可以分为过错缺陷和遗漏缺陷。如果把某些信息输入到了不正确的表现方式中，就称为过错缺陷；如果没有输入正确信息，就是遗漏缺陷。在这两种缺陷中遗漏缺陷更难检测和解决，但通过评审常常可以找出遗漏缺陷。

3. 失效（Failure）

在缺陷运行时，常常会发生失效的情况。一种是过错缺陷对应的失效，一种是遗漏缺陷对应的失效。在这两种失效类型中，遗漏失效是最难处理的，主要依赖有效的评审，发现遗漏缺陷来避免遗漏失效的产生。

4. 测试（Test）

测试是一项采用测试用例执行软件的活动，在这项活动中某个系统或组成的部分将在特定的条件下运行，然后要观察并记录结果，以便对系统或组成部分进行评价。测试活动有两个目标，即找出失效和显示软件执行正确。测试可能会由一个或多个测试用例组成。

5. 测试用例（Test Case）

测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。测试用例是执行的最小实体。

6. 回归测试（Regression Testing）

回归测试的目的是为了测试由于修正缺陷而更新的应用程序，以确保彻底修正了上一个

版本的缺陷，并且没有引入新的软件缺陷。回归测试可以采用手工测试或自动化测试来执行原来所报告的缺陷步骤和方法，检验软件缺陷是否被修正。回归测试又可分为：完全回归测试和部分回归测试。完全回归测试是对所有修正的缺陷进行验证。但由于测试时间紧张，需要验证的缺陷数量巨大，可以进行部分回归测试。

那么该如何使用二者呢？我们把测试用例按照测试优先级进行部分回归测试。将严重性高的缺陷进行回归测试。

1.3 软件测试技术分类

从不同的角度，可以把软件测试技术分成不同种类。

1. 从是否需要执行被测软件的角度分类

从是否需要执行被测软件的角度，可分为静态测试（Static Testing）和动态测试（Dynamic Testing）。

那些不利用计算运行被测程序，而是通过其他手段达到测试目的的方法称做静态测试。换句话说，就是计算机并不真正运行被测试的程序。如：在项目开发中存在着大量的规格说明，而规格说明是无法用计算机来运行的，所以对于这些软件的规格说明的测试就属于静态测试。除此之外，使用分析方法来进行的一些测试，如：对软件设计、体系结构和代码的审查也是静态测试。但这并不是说，静态测试完全脱离了计算机。实质上，静态测试有时候会利用计算机作为对被测试程序进行特性分析的工具，只是不真正运行被测试程序。

静态测试的方法主要有代码检查和走查，以及桌面检查和同行评分。其中，代码检查与走查是两种主要的人工测试方法，都要求组成一个小组来阅读或直观检查特定的程序。无论采用哪种方法，参加者都需要完成一些准备工作。所谓的准备工作就是组织参加者召开会议，会议的目标就是找出错误，但不必找出改正错误的方法。

代码检查与走查已经广泛运用了很长时间。在代码检查与走查中，一组开发人员（3～4人为最佳）对代码进行审核，参加者当中只有一人是程序编写者。也就是说这项工作应该主要是由其他人，而不是由软件编写者本人单独来完成。这种做法符合软件测试的原则，即软件编写者往往不能有效地测试自己编写的软件。

代码检查与走查是对以前的桌面检查过程（在提交测试前由程序员阅读自己程序的过程）的改进。二者相比，代码检查与走查更为有效，因为在该项工作的实施过程中，除了软件编写者本人之外，还有其他人参与。

代码检查与走查的另一个优点在于，一旦发现错误，通常就能在代码中对其进行精确定位，降低了调试（错误修正）的成本。另外，这个过程通常能够发现成批的错误，这样错误就可以一同得到修正。而基于计算机的测试通常只能暴露出错误的某种特征（程序不能停止，或打印出了一个无意义的结果），只能一个一个地发现并纠正错误。

在典型的程序中，这些方法通常会有效地查找出 30%～70%的逻辑设计和编码错误。但是，这些方法不能有效地查找出高层次的设计错误，例如在软件需求分析阶段的错误。请注意，所谓 30%～70%的错误发现率，并不是说所有错误中多达 70%可能会被找出来，而是讲这些方法在测试过程结束时可以有效地查找出多达 70%的已知错误。

当然，可能有人认为人工方法只能发现“简单”的错误（即与基于计算机的测试方法相