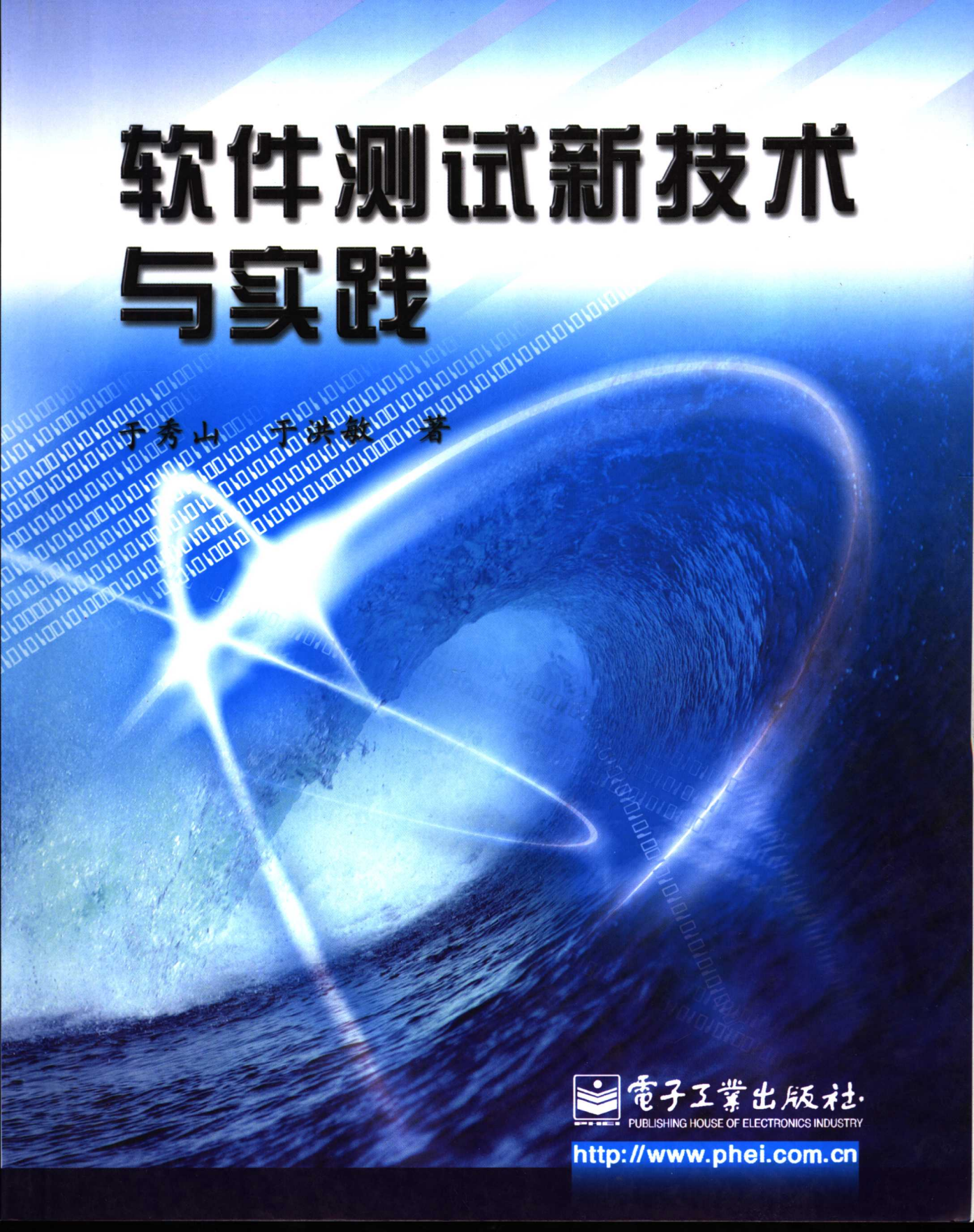
软件测试新技术与实践

于秀山 于洪敏 著



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

软件测试新技术与实践

于秀山　于洪敏　著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书系统地介绍了近几年出现的软件测试新技术，这些新技术包括正交试验设计、均匀试验设计、成对组合覆盖、被动测试、符号执行和数据驱动软件测试技术。

为了提高实用性，书中结合不同的软件项目，给出了应用实例。通过这些实例，读者可以容易地掌握这些技术并加以灵活应用。

本书是软件测试方面的中高级读物，适用于中高级软件测试人员和质量管理人员，同时对初级软件测试人员也有一定的帮助。本书还可以作为计算机专业大学生和研究生的参考书。

图书在版编目（CIP）数据

软件测试新技术与实践 / 于秀山，于洪敏著. —北京：电子工业出版社，2006.9
ISBN 7-121-03144-2

Ⅰ. 软… Ⅱ. ①于… ②于… Ⅲ. 软件－测试 Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2006）第 104061 号

责任编辑：张燕虹
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：10　字数：220 千字
印　　次：2006 年 9 月第 1 次印刷
定　　价：16.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系电话：（010）68279077；邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

反侵权盗版声明

前　言

当今世界，信息技术已成为国家经济发展的支柱产业之一，作为信息技术重要组成部分的软件产业取得了长足的发展。软件成为与人们日常生活息息相关的一部分。

软件测试是伴随着软件的产生而产生的。软件危机的频繁出现，使软件测试的地位得到了前所未有的提高。软件测试已经不仅仅是局限于软件开发过程中的一个阶段，它已经开始贯穿于整个软件开发过程，成为软件产品质量控制与质量管理的重要手段之一。美国质量保证研究所对软件测试的研究结果表明，越早发现软件中存在的问题，开发费用就越低。在编码后修改软件缺陷的成本是编码前的 10 倍，在产品交付后修改软件缺陷的成本是交付前的 10 倍。另外，根据对国际著名 IT 企业的统计，它们的软件测试费用占整个软件工程所有研发费用的 50%以上，软件测试时间占整个软件工程所有研发时间的 40%~50%。

随着软件工程学的发展和软件项目管理经验的提高，软件测试正在逐步成为一个新兴的产业。业内人士分析，该类职位的需求主要集中在沿海发达城市，北京和上海的需求量分别占 33%和 29%。其中，民企需求量最大，占 19%，外商独资欧美类企业需求量次之，占 15%。

高质量的软件要求，促进了软件开发技术的发展，同时也推动了软件测试技术的不断提高，新的软件测试技术相继涌现。本书结合软件测试项目，详细介绍了下面几种软件测试技术。

（1）组合测试技术：包括正交试验设计技术、均匀试验设计技术、成对组合覆盖技术。该技术主要解决测试用例设计中的参数组合问题，使得用较少的测试用例达到较好的测试效果。

（2）被动测试技术：该技术主要根据给定的输入/输出序列和软件行为模型，判断软件运行是否正常。该技术可广泛地用于网络监控等领域。

（3）符号执行测试技术：符号执行测试技术是软件静态分析中采用的一种技术。使用该技术，可以准确地确定覆盖某些软件指定路径的测试用例，并找出不可达的路径。

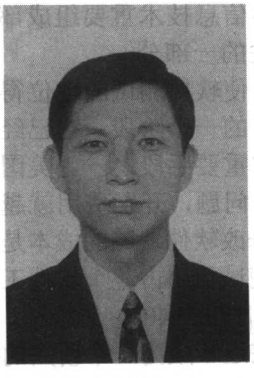
（4）数据驱动软件测试技术：主要介绍了解决如何通过 Excel 电子表格实现测试数据的存取、测试结果写入以及如何通过黑盒测试工具 WinRunner 编写数据驱动测试脚本。该技术可用来对软件进行自动测试、压力测试、容量测试等。

在本书编写期间，加拿大渥太华大学的 Alan W. Williams 博士、解放军南京理工大学博士研究生洪宇、同事董昕给予了一定的帮助，在此表示感谢。

鉴于作者水平有限，书中不乏遗漏和错误，敬请各界同仁斧正。

于秀山　于洪敏

作 者 介 绍



于秀山，男，1962 年 12 月出生，山东栖霞人，博士。现为总参第六十一研究所博士后，高级工程师，解放军南京理工大学硕士研究生导师，总参通信部科技创新工作站进站人员。

自 1998 年从事软件测试工作以来，完成了多项军队软件测试方面的科研项目，承担了多项大型软件项目的测试工作，获军队科技进步二等奖 1 项，三等奖 1 项；获人防科技进步二等奖 1 项，三等奖 1 项。发表论文 20 篇，出版译著 13 部。

2004 年 8 月至 2005 年 8 月在加拿大渥太华大学贝尔实验室工作，研究方向为软件测试和软件质量保证。

于洪敏，男，1963 年 12 月出生，山东栖霞人，军事学博士，上校军衔。曾 3 次荣立三等功。现任军械工程学院装备指挥与管理系副主任、副教授。

1991 年以来，先后主持或参与 17 个科研课题的研究工作，有 12 个项目获得军队科技进步三等奖以上奖励，其中有 5 项排名第一；发表学术论文 60 余篇，其中，中文核心期刊、军事学重点期刊或统计源期刊 37 篇，国际会议 8 篇，有 4 篇被国际三大检索收录；主编或参与编写著作、教材、论文集 12 部；被评为军械工程学院优秀教员、科技工作先进个人、学科建设先进个人各 1 次，2 次荣获军队优秀人才岗位津贴三等奖。

目　录

第1章 概 论

随着软件应用范围和规模的不断扩大，软件设计的复杂程度不断提高，软件开发中出现错误或缺陷的机会越来越多。同时，人们对软件质量的要求越来越高，特别是一些涉及人身生命安全的应用领域，如航空航天、核电站等，其中的软件质量受到了空前的关注。作为软件质量保证手段之一的软件测试越来越受到重视，很多软件开发机构和软件质量管理部门都建立了独立或半独立的软件测试部门，软件测试从业人员大量增加，软件测试已成为一个新兴的行业。

本章首先采用分类的方法对软件测试技术进行简要的回顾；然后，介绍几种软件测试过程模型，并结合工程实践，给出一个软件测试过程模型；最后，介绍软件测试中的组合问题。

1.1 引 言

自20世纪60年代出现软件危机以来，软件从业人员、专家和学者为解决软件危机做出了大量的努力。人们已经逐步认识到软件危机是由软件中存在的缺陷（错误）而造成的，正是这些缺陷（错误）导致了软件开发在成本、进度和质量上的失控。软件中存在缺陷是难免的，问题在于我们如何去避免缺陷的产生和消除已经产生的缺陷，使程序中的缺陷密度达到可以承受的程度。为了更好地解决这些问题，软件界做出了各种各样的努力和尝试。

人们曾经认为更好的程序设计语言可以消除软件中的缺陷，由此推动了程序设计语言的发展。为了使程序更易于理解，开发了结构化程序设计语言；为了满足实时多任务需求，开发了结构化多任务程序设计语言；为了提高重用性，开发了面向对象的程序设计语言；为了避免需求理解的二义性，开发了形式化描述语言；为了支持大型数据库应用，开发了可视化工具。程序设计语言对提高软件生产效率起到了一定的积极作用，但对整个软件质量尤其是软件可靠性的影响，与其他因素相比，作用发挥得不明显。

人们也曾经试图用形式化证明方法来证明程序的正确性。数学家对形式化证明方法最感兴趣，他们将程序作为数学对象，采用数学方法证明程序的正确性，并发表了很多相关的论文，但实际作用却非常有限，因为在形式化需求描述语言不完善的情况下，形式化证明方法只有在代码完成之后才能使用，而且对于一些大的程序，几乎无法证明其正确性。

受其他行业项目工程化的启发，软件工程学出现了。软件开发被视为一项工程，采用工程化的方法来计划和管理软件的开发工作。在软件工程中，软件已不仅仅是指程序了，软件包括程序以及开发、使用、维护程序所需的所有文档、数据。研制软件也不仅仅是编

程序了，编码只是整个软件研制过程中的一个部分，并将软件测试也作为其中的一个过程。

针对需求不确定的问题，出现了不同的软件开发模型，如瀑布模型、V 模型、螺旋模型、增量模型、渐进模型、净室过程、快速应用程序开发（RAD）、协同应用程序开发（JAD）以及 Rational 统一过程（RUP）等。

事实上，对于软件来讲，还没有像银弹那样的东西。不论采用何种技术和方法，软件中仍然会有缺陷。采用新的语言、先进的开发方式、完善的开发过程，可以减少缺陷的产生机会，但是不可能完全杜绝软件中的缺陷，软件测试是目前查找这些缺陷的有效方法。统计表明，在典型的软件开发项目中，软件测试工作量占软件开发总工作量的 40%以上。在软件开发的总成本中，软件测试开销占 30%~50%。如果把维护阶段也考虑在内，测试的成本比例也许会有所降低，但实际上维护工作相当于二次开发，乃至多次开发，对于每一次开发，都需要进行相应的回归测试工作。

近 10 年来，软件测试理论得到了快速发展，出现了很多新的软件测试方法和测试工具，软件测试技术已经发展成为一门独立的学科。

1.2 软件测试技术分类

软件开发技术的发展推动了软件测试技术的发展，出现了各种不同的软件测试技术，对于这些技术，可以从不同的角度加以分类。

（1）从是否需要执行被测软件的角度，可分为静态测试和动态测试。

（2）从测试是否针对系统的内部结构和具体实现算法的角度，可分为白盒测试和黑盒测试。

（3）从测试执行时是否需要人工干预的角度，可分为人工测试和自动测试。

其中，每一种技术还可以继续进行分类，如图 1.1 所示。

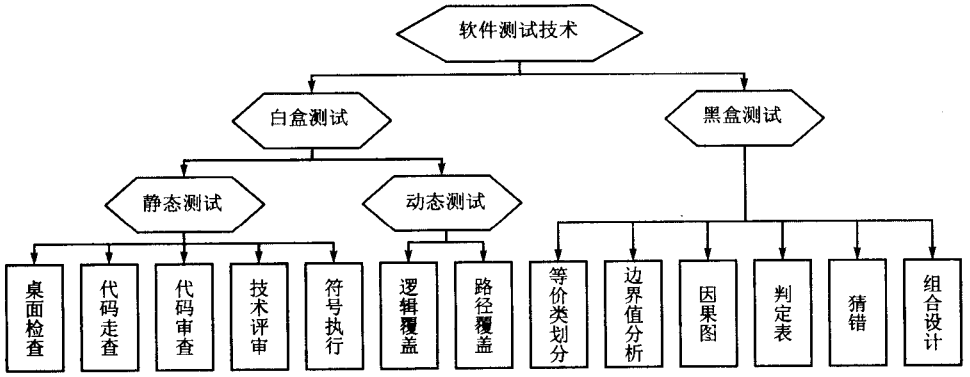


图 1.1 软件测试技术分类

需要指出的是，这些方法之间存在交叉。例如，动态测试也可分为黑盒测试和白盒测试，静态测试一般只包含白盒测试；黑盒测试一般都是动态测试，而白盒测试一般包括动态测试和静态测试。软件测试工具的发展，如白盒测试工具 Logiscope、黑盒测试工具 WinRunner 等，使得很多以往靠人工完成的测试工作可以通过工具自动或半自动完成。

1.2.1 黑盒测试

黑盒测试也称功能测试，主要从功能的角度检测被测系统是否满足软件需求规格说明的要求。这种测试方法把程序看成一个不能打开的黑盒子，着眼于程序外部结构，不考虑程序内部逻辑结构和内部特性，只考虑被测系统提供的用户界面和其他外部接口特性。通过分析用户界面的特性（字段类型、取值范围等）以及外部接口的数据格式，选择覆盖每一个功能的测试数据，并通过被测系统提供的用户界面和其他外部接口输入测试数据，驱动被测系统运行，通过检测系统的输出判定被测系统是否满足需求规格说明的要求。由于软件输入空间的无限性，使得不可能也没有必要遍历软件的所有输入。黑盒测试的目标是用尽可能少的测试输入，覆盖尽可能多的软件需求（功能）。为了达到这个目标，人们提出了各种不同的黑盒测试方法，主要有等价类划分、边界值分析、因果图、组合设计、错误猜测等。在实际应用中，通常采用测试用例对系统功能点的覆盖程度衡量黑盒测试的充分性。

1.2.2 白盒测试

白盒测试也称结构测试或逻辑驱动测试，主要用来找出程序内部的逻辑错误或不可达路径。在测试时，需要知道程序内部的逻辑结构及具体的代码实现，并通过分析程序的内部结构，找出覆盖不同路径的测试输入。白盒测试的目标是用尽可能少的测试输入，覆盖尽可能多的程序语句或路径。白盒测试包括静态和动态两大类。静态方法主要包括桌面检查、代码走查、代码审查、技术评审、符号执行；动态方法主要包括逻辑覆盖和路径覆盖。

软件的功能是靠程序实现的，功能是程序的外在表现形式，覆盖了一定功能集的黑盒测试用例集一定能够覆盖与此功能集对应的程序。同样，程序是功能的具体实现方式，覆盖了一定程序的白盒测试用例一定能够覆盖它所实现的程序功能。白盒测试和黑盒测试从 2 个不同的侧面出发设计测试用例，各有侧重，要结合使用。图 1.2 给出了动态测试过程。

在动态测试过程中，测试用例来源于 2 个不同的方面：一是采用黑盒测试方法，通过分析软件规格说明书产生；二是采用白盒测试方法，通过分析被测软件源程序代码产生。然后，用测试用例产生的测试数据驱动被测软件执行，得到实际的输出结果，将这个结果与通过软件规格说明书得到的期望值进行比较，判断两者是否一致。

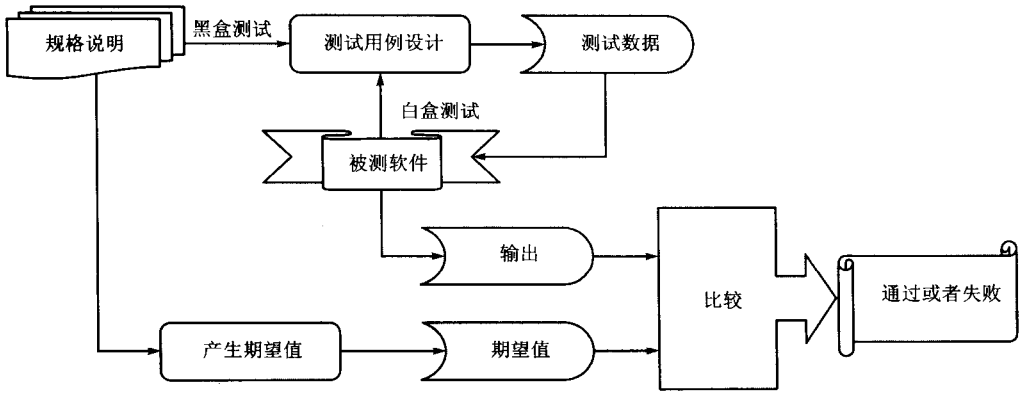


图 1.2　动态测试过程

1.2.3　软件测试策略

不同的测试方法有其自身的特点和适用场合，同时，测试方法的评价也有不同的标准，下面从故障检测效率、故障检测开销和检出的故障等级 3 个不同的方面对有关测试方法做一比较。

1．静态测试和动态测试比较

对典型软件的测试试验结果表明，采用代码审查和代码走查方法能够发现 30%~70%的逻辑设计和编码错误，如果采用更正式的数学方法，能够发现程序中 90%的错误。就错误检测而言，静态测试比动态测试更加有效和经济。

Basili 和 Selby 选择了 4 个不同的软件系统和一批有经验的程序员对静态测试、动态测试进行了比较，结果表明，一个专业的程序员通过阅读代码发现的故障数量比功能测试和语句覆盖测试多，故障检测率也高。功能测试发现的故障比语句覆盖测试多，但两者的故障检测率相同。静态测试效果对测试人员水平的依赖性较大。代码阅读能够发现更多的接口方面的故障，而功能测试能够发现更多的控制方面的故障。

2．黑盒测试和白盒态测试比较

Poston 和 Sokolsky 应用黑盒测试和白盒测试方法对典型的三角形程序进行了测试，得到如下结果：

（1）用白盒测试方法设计的测试用例能够百分之百地实现语句、分支和控制流覆盖，因此，从代码覆盖的角度来看，其覆盖率最高，但这些测试用例没有找出代码中存在的 4

个缺陷，因此，其故障检测能力低。

（2）用黑盒测试方法设计的测试用例能够百分之百地实现需求覆盖，因此，从需求覆盖的角度来看，其覆盖率最高。这些测试用例同时实现了百分之百的语句、分支和控制流覆盖，因此，其代码覆盖率也达到最高，此外，黑盒测试还发现了白盒测试没有发现的缺陷。

3. 测试策略

根据以上研究结果，结合软件开发过程的不同阶段，建议采取下面的测试策略。

（1）先进行静态测试，后进行动态测试。

（2）开发文档和源程序可应用静态方法，通过人工和静态分析工具实施。

（3）单元测试可应用白盒测试方法，采用人工和单元测试工具相结合的方法实施。

（4）集成测试应用灰盒测试方法。对于接口参数之间的组合，可以首先通过等价类划分、边界值分析确定出有代表性的测试数据，然后采用因果图、组合设计等方法选出有代表性的参数组合，最后通过猜错法、逻辑覆盖以及路径覆盖等方法补充测试用例。以上测试既可以人工执行，也可以借助集成测试工具执行。

（5）系统测试和确认测试应用黑盒测试方法。对于输入界面的字段，首先，采用等价类划分、边界值分析确定出有代表性的测试数据；然后，采用因果图、组合设计等方法选出有代表性的参数组合；最后，通过猜错法以及其他方法补充测试用例。对于重要的软件或功能，还要采用逻辑覆盖以及路径覆盖等方法对其进行进一步测试。以上测试既可以人工执行，也可以借助黑盒测试工具执行。

（6）对于特殊类型的测试，如可靠性测试、压力测试、性能测试等，在采用上述方法测试后，还要针对测试要求，根据软件的运行剖面，如随机分布、均匀分布、泊松分布等设计测试用例。

1.3 测试过程模型

软件测试是由一系列活动组成的，软件测试过程模型用于定义软件测试的流程和方法。众所周知，开发过程的质量决定了软件的质量，同样，测试过程的质量将直接影响测试结果的准确性和有效性。软件测试过程与软件开发过程一样，都遵循软件工程原理，遵循管理学原理。

随着测试过程管理的发展，软件测试专家通过实践总结出了很多很好的测试过程模型。这些模型对测试活动进行了抽象，并与开发活动有机地进行了结合，是测试过程管理的重要参考依据。

1.3.1 基于工作流程的模型

软件测试是与软件开发紧密相关的一系列有计划的系统性活动，不同的开发模型要求具有不同的测试过程。下面结合软件开发模型介绍 3 种软件测试过程模型。

1. V 模型

V 模型最早是由 Paul Rook 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 模型是最具有代表意义的测试模型，它是软件开发瀑布模型的变种，反映了测试活动与分析和设计的关系，如图 1.3 所示。

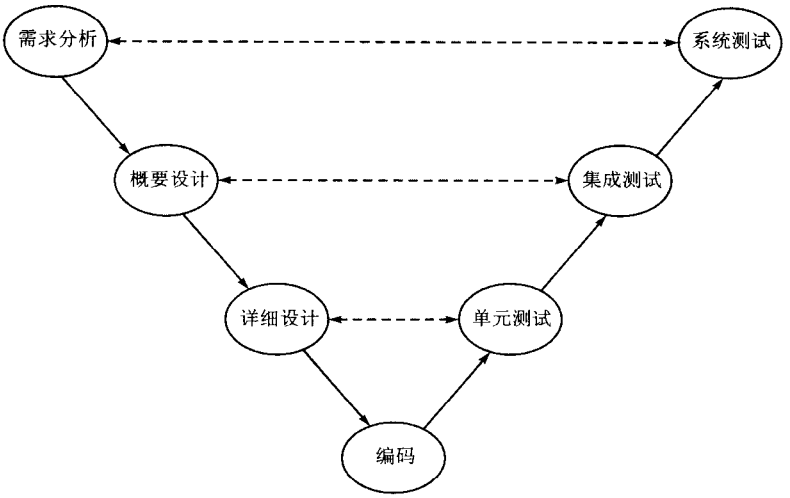


图 1.3 软件测试过程 V 模型

V 模型清晰地展示了动态测试的全部过程，并定义了动态测试与开发之间的关系。在图 1.3 中，左半部分描述了基本的开发过程，右半部分描述了与开发过程对应的测试过程。V 模型认为动态测试的行为与开发过程的行为是相对应的，其每一阶段的基础就是相对应开发阶段的文档，如单元测试的基础是详细设计文档，集成测试的基础是概要设计文档。

在实际应用中，V 模型有下述 3 个致命的缺陷。

（1）测试与开发文档之间很少有这种完美的一对一关系。例如需求分析的文档不足以为系统测试提供足够的信息，系统测试通常还需要概要设计文档与详细设计文档中的一部分内容。

（2）最重要的是，V 模型完全没有提及静态测试，忽略了代码审查、需求测试等重要

的测试手段，它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，主要是通过测试程序来查找错误，这样，在前期阶段隐藏的问题一直要等到后期的测试才可能被发现，为了修改这些错误，往往需要花费很大的代价。

（3）软件测试时间经常由于前期开发阶段进度的拖延而被挤占，甚至取消，使测试质量大打折扣。

2. W 模型

针对 V 模型的缺陷，Paul Herzlich 在 1993 年最早提出了 W 模型，如图 1.4 所示。

W 模型由 2 个 V 字形模型组成，分别代表测试与开发过程，开发过程位于图的左半边，测试过程位于图的右半边。从图 1.4 可以看出，每一个开发行为都对应着一个测试行为，开发行为若是各种需求的定义与编写，相对应的测试行为则是对这些文档的静态测试；开发行为若是软件的实现，相对应的测试行为则是对这些实现的动态测试。W 模型明确表示出了测试与开发的并行关系，充分体现了测试贯穿于整个开发过程中的思想。

相比于 V 模型，W 模型更科学。W 模型可以说是前者自然而然的发展，W 模型不仅强调测试应伴随着整个软件开发周期，而且指出测试的对象不仅仅是程序，需求、设计等文档同样需要测试。W 模型体现了“尽早地和不断地进行软件测试”的原则。例如，需求分析完成后，测试人员就应该参与到需求的验证和确认活动中，以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，及早制定应对措施，这将显著减少总体测试时间，加快项目进度，减少项目开销。

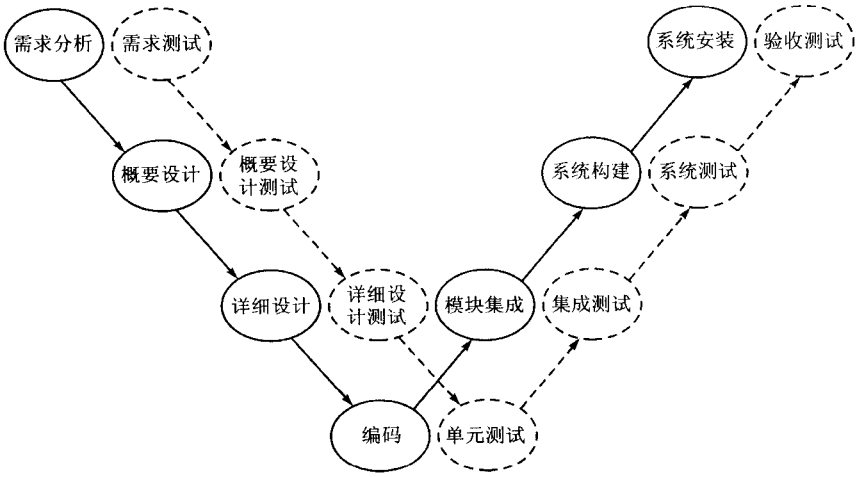


图 1.4 软件测试过程 W 模型

V 模型与 W 模型有一个共同点，即开发行为与测试行为是一一对应的。但是，W 模

型并不主张动态测试必须要与开发阶段对应起来，在某些情况下，例如有系统测试与验收测试中的功能测试、安全性测试、性能测试、用户验收测试就可以构成动态测试的全部内容。同时，W 模型也不限制动态测试行为必须严格地基于对应开发行为产生的单一文档。

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的，同时，测试和开发活动也保持着一种线性的前后关系，只有上一阶段完全结束，才可正式开始下一个阶段工作，从而无法支持迭代的软件开发模型。

3. H 模型

如前所述，V 模型和 W 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，而事实上，这些活动在大部分时间内是可以交叉进行的，特别是相对于迭代软件开发模型，情况更是如此。因此，相应的开发与测试之间不再存在严格的次序关系。同时，单元测试、集成测试、系统测试等也存在反复和交替，不一定始终保持前后顺序关系。

为了解决以上问题，有专家提出了 H 模型。H 模型将测试活动视为一个完全独立的活动，具有自己完全独立的流程。该流程主要包括测试准备活动和测试执行活动，如图 1.5 所示。

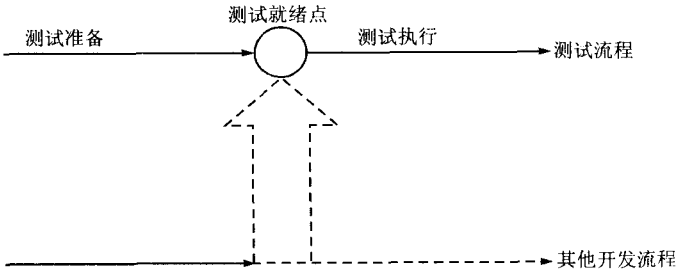


图 1.5　软件测试过程 H 模型

图 1.5 说明了在软件开发某个阶段所对应的测试活动。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。也就是说，只要测试准备就绪了，就可以开始测试执行活动。在整个软件生命周期内，存在多个这样独立的测试流程，这个流程与其他流程并发地进行。H 模型也包含了软件测试要尽早准备、尽早执行的思想，并且，不同的测试活动可以按照某个次序先后进行，也可以按照其他的顺序展开，哪个阶段的测试活动达到测试就绪点，测试执行活动就可以开始。

1.3.2 测试成熟度模型

1996年，美国伊利诺伊州工学院（Illinois Institute of Technology）的Ilene Burnstein等人参照CMM提出了测试成熟度模型（Testing Maturity Model，TMM），并将其作为CMM的补充。该模型用来评价软件测试机构的测试能力，以便于进一步改进软件测试过程。

TMM将测试能力分为下述5个等级。

（1）Level 1——初始级。软件测试是一个完全混乱的过程，测试与调试相互交叉，软件开发过程中缺乏测试资源、测试工具以及训练有素的测试人员，缺乏成熟的测试目标，测试处于可有可无的地位。

（2）Level 2——阶段定义级。已具备基本的测试技术和方法，测试已经与程序调试区分开来，并且被定义为紧随软件编码完成之后的一个独立的阶段，测试的首要目的就是验证软件是否符合需求。

（3）Level 3——集成级。测试不再仅是编码完成之后的一个阶段，它已经被集成到整个软件生命周期中，测试活动遵循软件生命周期V模型。

（4）Level 4——管理和度量级。测试过程不再仅是定性的描述，它已经是一个可量化和可度量的过程。测试活动除测试被测程序外，还包括软件生命周期中各个阶段的评审和审查。

（5）Level 5——优化、缺陷预防和质量控制级。在前4级的基础上，已经建立起测试规范和流程，测试是受控的和被管理的。已经具备了调整、连续改进过程的机制以及缺陷预防和质量控制机制。

TMM的可操作框架提供了一系列的等级层次。这些等级层次包括成熟度目标、子目标、活动、任务等。通过评价机构的测试能力所处的成熟度层次可以定义其可信度。TMM确定了一些关键过程域（Key Process Area，KPA），可以通过改进这些过程域来改善测试机构的测试过程。

Burnstein提出的5级成熟度模型为测试过程能力评估和测试过程改进提供了参考，但在其模型中还没有提出具体的可操作的改进方法，并且5级模型在很大程度上还只是定性的理论框架性质的描述，没有形成可操作的定量化的操作规程。

1.3.3 软件测试过程模型选取策略

前面介绍的测试过程模型中，V模型强调了在整个项目开发中需要经历的不同的测试阶段，但忽视了测试对象不应该仅仅是程序。W模型对此进行了补充，明确地指出应该对