第 1 章

软件测试概述

本章要点：

- 软件缺陷及其产生原因。
- 软件测试的目标。
- 软件测试中的认识误区。
- 软件测试的发展史。
- 软件测试的国内外现状。
- 软件测试的发展趋势。

软件是与计算机系统操作相关的程序、数据和文档，是人类社会高度发展的产物，是人类智慧的结晶。从 20 世纪 50 年代初，软件技术不断取得进展，使软件在内涵、规模、开发方法、应用领域等方面都发生着日新月异的变化。软件越来越多地影响和改变人类生活的各方面。然而，软件构成及开发的日益复杂、软件应用领域的日益拓宽也使得人们常常受到有缺陷的软件的影响，软件缺陷给人们带来了许多物质上和精神上的损失。软件质量不断受到人们的重视，为了发现软件中的缺陷，保证软件质量，软件测试应运而生。本章主要介绍软件测试的目标、发展史、现状及发展趋势，以及人们对软件测试的认识误区，为后续章节的学习打下基础。

1.1 软件测试的意义

1.1.1 软件缺陷的典型例子

俗话说，人无完人，金无足赤。人类智慧的结晶——软件也难以尽善尽美，存在缺陷的事情常有发生，可将软件中的所有质量问题都称为软件缺陷。软件缺陷可小可大，然而软件的性质决定了即使是一个很小的缺陷，也很有可能给使用者带来极大的损失。下面就是一些造成较大影响的软件缺陷的例子。

1. 千年虫问题

20 世纪 70 年代，当时计算机的存储空间很小，程序员为了缩减程序所占的存储空间，将表示年份的 4 位数中的前两位去掉，如“1985”被表示为“85”。

这些程序员当然知道到了2000年这种表示方法会带来麻烦，然而，他们认为不会等到2000年，软件系统就应该更新换代了，所以并没有顾及未来的问题。

不幸的是，这些程序员采用的年份表达方法在2000年快要到来时仍被广泛使用，这时千年虫问题才引起世界各国众多行业尤其是银行业、零售业、电信业的高度重视。为解决千年虫问题，避免出现如2001年与1901年在计算机日期表达中的混淆不清，全世界已付出数千亿美元的代价。

2. 爱国者导弹中的软件缺陷

在第一次海湾战争中，美军最大的一次伤亡是1991年2月25日在沙特阿拉伯的宰赫兰兵营被伊拉克的飞毛腿导弹击中，死28人，伤98人。其原因是，大名鼎鼎的爱国者导弹的软件中存在缺陷，导弹运行100小时共形成343.3毫秒的积累误差，导致687米的距离偏差，因此未能成功地拦截飞毛腿导弹。

有意思的是，发现这一微小偏差的并不是爱国者导弹的设计者——美国人，而是思维严密的以色列人。

3. 迪斯尼的圣诞节礼物

1994年圣诞节前夕，迪斯尼公司发布了第一个面向儿童的多媒体光盘游戏—“狮子王童话”。这是迪斯尼公司第一次进军儿童计算机游戏市场，由于该公司的品牌效应以及大力的广告宣传，“狮子王童话”的市场销售情况非常好，该游戏成为大多数父母圣诞节为孩子必买的礼物。

但随后的情况却出人意料。12月26日，很多客户反映该游戏在自己的机器上无法成功安装或无法正常使用。后来才证实，出现这种情况的原因是迪斯尼公司没有针对“狮子王童话”可能使用的各种机型进行系统兼容性测试，只是对少数机型进行了兼容性测试，所以导致该款游戏只能在少数几种机器上成功安装和运行。

4. 阿丽亚娜火箭发射失败

1996年6月，欧洲阿丽亚娜5型火箭第一次发射，由于定位软件出错，导致计算机命令固态推进器与主发动机尾喷管发生偏离，结果火箭发射升空仅40秒就爆炸了。

5. “冲击波”计算机病毒

2003年8月，“冲击波”计算机病毒首先在美国发作，导致美国政府机关、企业和个人成千上万台计算机受到攻击。随后，“冲击波”蠕虫病毒很快在因特网上广泛传播，中国、日本、欧洲等地的用户也受到了攻击，结果是大量的邮件服务器瘫痪，给整个世界范围内的网络通信带来了惨重的损失。

制造“冲击波”病毒的黑客只用了3周时间就完成了该病毒程序。该病毒仅仅利用微软公司Messenger Service中的一个缺陷，就攻破了计算机安全屏障，使所有基于Windows操作系统的计算机崩溃。更令计算机安全专家担忧的是，如不立即采用有效的防御措施，黑客将很快找到利用该缺陷控制大部分计算机的方法。

随后，微软公司紧急发布了升级补丁，以修复操作系统中存在的缺陷，抵御该病毒的攻击。

6. 微软 64 位服务器软件缺陷

2004 年上半年，微软公司承认，如果客户使用的是 64 位的 Windows Server 2003 企业版，并且硬件配置是英特尔安腾芯片，很可能突然死机。更可怕的情况是，死机后根本不可能重新启动，将给企业带来巨大的损失。

微软公司称，该问题是由硬件管理程序在检测硬件设备时造成的，随后即推出了升级补丁程序。

7. 索尼电视软件缺陷

2006 年 2 月，索尼(中国)公司称，2005 年下半年在中国内地推出的 5 款电视，包括液晶电视和液晶背投电视，由于在软件方面出现了设计缺陷，导致不能正常开关机。

索尼公司的专业人员研究后发现，特定范围内的液晶背投电视和液晶电视的软件中存在一个计时错误，该错误会导致相关型号电视在待机及累计工作约 1200 小时后，出现在使用中不能正常关机或在待机状态下不能开机的现象。而液晶电视的正常工作时间为 5 万小时。随后索尼(中国)有限公司对存在问题的 5 款电视进行了免费软件升级。

1.1.2　软件缺陷的产生原因

软件缺陷给人类带来的损失，是刚才的几个例子远远不能概括的。这种损失有经济上的、精神上的、身体上的，更有许多人因为软件缺陷而付出了生命的代价。

软件缺陷产生的原因到底是什么呢？软件产品是整个软件开发活动的成果结晶，所以我们应该到软件开发活动的各阶段去寻找原因。

按照软件工程的观点，在对软件系统的可行性进行论证之后，软件开发的主要阶段依次为需求分析、软件设计、软件编码、软件测试、软件运行和维护，如图 1-1 所示。据众多软件从业人员的亲身项目实践，得出的结论是，软件需求分析不够全面、准确是导致软件缺陷的最主要原因。

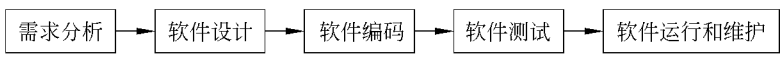

图 1-1　软件开发的主要阶段

需求分析的主要任务是确定待开发软件的功能需求、性能需求及运行环境约束。简单地说，需求分析是确定待开发软件“做什么”。由于软件开发最终是要交付用户使用的，因此所谓“需求”是指用户对软件的需求。系统分析人员和开发人员应在与用户反复、充分沟通的基础上完成需求分析。

由于软件是复杂的逻辑产品，用户在最终拿到软件之前往往很难一次性地精确描述对软件的需求，再加上系统分析人员、开发人员、用户对软件需求的关注角度和描述方式的不同，使得对需求的全面、准确的理解往往不能在需求分析阶段一蹴而就，而是随着软件开发活动的进行不断得到深化。需求分析阶段确定的需求不全面、不准确，即为软件缺陷的产生埋下了祸根。

软件设计和编码过程中的失误也会导致软件缺陷的产生，例如软件设计阶段考虑问题的片面性，软件设计文档不够具体，编码阶段的错误等。但很多情况下，不正确的软件

设计是不正确的需求分析引起的，编码阶段出现的错误则是由需求分析和软件设计不够完善、准确引起的。

1.1.3 软件测试的目标

尽管软件缺陷产生的原因已为很多人所知，但由于软件本身以及软件开发活动的特点，软件缺陷很难根除。所以，在软件交付使用前，为了尽量消除软件中的缺陷，对其进行测试是必不可少的。

软件测试的目标在早期被认为是尽可能多地发现软件中的潜在错误，这一点也可以从 Glenford J. Myers 给出的以下关于软件测试的规则描述中看出。

- 测试是为了发现程序中的错误而执行程序的过程。
- 好的测试方案是尽可能发现迄今为止尚未发现的错误的测试方案。
- 成功的测试是发现了迄今为止尚未发现的错误的测试。

当前仍然有部分人对软件测试存在误解，他们认为软件测试就是要证明软件是正确、可用的，能够满足用户的需要，而不是尽可能多地暴露软件中的潜在错误。

首先，这种想法是行不通的。由于软件是一种复杂的逻辑产品，对软件进行穷举测试是不可能的。因此，即使到目前为止对一个软件的测试中未发现任何错误，也不能说明该软件是绝对正确的，正所谓“软件测试只能证明软件有错，不能证明软件无错”。

从另一个角度说，对软件测试存有这种误解的人在进行测试时，往往在心理上会忽略软件中可能存在的缺陷，而把注意力集中在软件能否完成基本的、已知的功能上，这样的测试显然不是成功的测试。

1983 年，在 Glenford J. Myers 观点的基础上，Bill Hetzel 指出，软件测试的目标不仅是尽可能多地发现软件中的错误，还要对软件质量进行度量和评估，以提高软件质量。这一论断将对软件测试的认识提升到更高的层次。

1983 年，IEEE 对软件测试的定义则指出，软件测试的目标是为了检验软件系统是否满足用户的需求。

1.2 软件测试中的认识误区

软件测试的概念处于不断的发展之中，再加上缺乏软件测试的理论知识和实践经验，使得一些人对软件测试的认识存在误区。以下列举对软件测试的一些典型认识误区。

误区 1 测试和调试是一回事。

测试和调试有着根本的不同。测试是一个有计划、可重复的过程，目的是为了发现软件中的潜在错误和缺陷；而调试是一个随机的、不可重复的过程，目的是寻找错误的原因和具体位置，并修复错误。调试一般在测试后进行，当然，调试之后很可能又要进行测试，所以两者常交叉进行。

误区 2 可以对软件进行穷举测试。

对软件进行穷举测试是不可能的。也就是说，不可能对软件进行完全的测试以发现软件中的所有错误和缺陷。

这主要是因为，由于需求规格说明的复杂性和程序逻辑的复杂性，如下的测试是难以

做到的。

- 测试程序中的所有输入条件的取值。
- 测试程序中所有输入条件取值的组合。
- 测试程序中的所有路径。
- 测试出程序中所有潜在的错误和缺陷，例如由于需求分析不完善而导致的错误。

误区 3　若交付使用的软件有缺陷，是测试人员的失职。

软件测试的主要目标是发现软件中更多的错误和缺陷，但不能通过已有的测试证明某一个软件是绝对正确的。

交付使用的软件有缺陷，与多方面的人员都有关系，如系统分析人员、设计人员、编码人员、测试人员。软件开发中的任何一个环节出现问题，都有可能使软件出现缺陷。当然，这并不是说软件测试人员可以推卸责任，开发团队中多方面的人员加强沟通、合作才是最重要的。

误区 4　关注测试的执行而忽略测试用例的设计。

测试用例(Test Case)是为特定目标开发的一组测试输入、执行条件和预期结果，其目的是测试程序中的某路径，或核实程序或软件是否满足某个特定的需求。

从一定意义上说，设计测试用例是软件测试活动中具有核心地位的一个环节。若不注重测试用例的设计，很可能会遗漏有价值的测试用例，或导致设计出的测试用例不够准确，从而不可能对软件进行充分、有效的测试。

误区 5　测试比编程容易得多。

从某种意义上讲，对软件测试人员的要求比编程人员的要求高。这是因为，测试是一件十分复杂的工作。测试人员应具有细致沉稳的性格，很强的专业素质，对被测试软件的功能及架构十分清楚；测试人员还要能在无法实现穷举测试的前提下编写若干有价值的测试用例，以尽可能地揭露软件中的错误和缺陷；在进行自动化测试时，测试人员还应有编写测试脚本的能力。

误区 6　测试是编码之后进行的工作。

从软件开发的瀑布模型，人们容易得出这一错误的结论。事实上，软件生命周期中的测试阶段只是表明该阶段的主要工作是测试，并不意味着测试工作不能在需求分析阶段、设计阶段和编码阶段进行。

软件测试不仅是对程序的测试，需求分析和设计也应成为测试的对象。大量测试实践表明，软件中的大部分错误是在编码之前造成的，需求分析和设计造成的软件错误约占所有错误的 63%，而编码造成的错误仅占 37%。软件中的错误被发现得越晚，为修复它所付出的代价就越大。

所以在软件开发过程中，应尽早地、全面地开展测试。软件测试应成为一个独立的流程，可贯穿到软件开发的其他各流程，如需求分析、设计、编码，并与之并发地执行。在某流程中，达到恰当的测试就绪点即可进行独立的测试工作。测试还应是可迭代的。只有做到这些，测试才可能是成功的。

误区 7　测试自动化是万能的。

软件测试自动化可以提高测试的效率，但成本较高，需要自动化测试工具，还需要测

试人员编写测试脚本等。因此，只有需要经常执行的测试用例才适合于自动化测试。

当对某软件的测试自动化达到一定程度时，再想提高其自动化程度将会变得十分困难，需要付出很大的成本。2/8 原则同样适用于测试自动化，即付出 20%的成本可以实现 80%的测试自动化，若要实现剩余的 20%测试自动化，还需付出 80%的成本。所以在开发一个软件的过程中，不应盲目地提高其测试自动化的程度。

误区 8 软件测试是一种破坏性工作。

一些人认为，以尽可能发现软件中的错误为主要目标的软件测试是一种破坏性工作，是对软件开发人员工作的否定。这种想法在很大程度上制约了软件测试的发展，且十分不利于软件质量的保证和提高。

1.3 软件测试的发展史及现状

1.3.1 软件测试的发展史

软件测试是伴随着软件开发活动的产生而产生的。早在 20 世纪 50 年代，英国著名的计算机科学家图灵就给出了软件测试的原始定义。图灵认为，软件测试是程序正确性证明的一种极端的实验形式。这导致人们对软件测试的理解较为狭隘。

当时的软件测试主要针对用机器语言和汇编语言编写的程序，通过设计并运行测试用例，将运行结果与测试人员的预期结果进行比较，从而判断程序的正确性。人们在设计测试用例时常凭经验或直觉，这使得测试用例具有不完备性，测试也不够充分和有效，通过测试之后的软件往往还隐藏着大大小小的缺陷。

当时还有很多人认为，测试就是“调试”，是查找软件中已发现错误的原因并纠正错误的一种活动。人们对软件测试不够重视，在测试上投入的人力、物力也很少，而且测试介入开发过程的时间点很晚，往往是在软件的编码结束后才进行测试。

直到 1957 年，人们才认识到测试应该是一种发现软件中潜在错误和缺陷的活动。1972 年，在北卡罗来纳大学举行了首届软件测试正式会议。1975 年，John Good Enough 和 Susan Gerhart 发表了名为《测试数据选择的原理》的文章，使软件测试得到了许多研究者的重视。1979 年，Glenford J. Myers 的著作《软件测试的艺术》(*The Art of Software Testing*)可以说是软件测试领域一本最重要的专著。Glenford J. Myers 在书中对软件测试的定义、目标等进行了描述，他认为测试是为发现错误而执行程序的过程，测试的目的在于尽可能多地发现软件中的错误。

20 世纪 80 年代早期，软件质量日益得到人们的重视。1983 年，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。”

1983 年，IEEE 提出的软件工程术语指出：“测试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，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。”这个定义指出了软件测试的目的是检验软件系统是否满足需求。

进入 20 世纪 90 年代，软件测试得到了快速发展。随着面向对象分析和设计技术的普遍应用，面向对象的软件测试日益受到人们的重视，面向对象的软件测试的理论和技术

不断被完善；各种软件测试工具不断被开发出来，使软件测试的自动化程度不断提高；越来越多的测试模型被提出，测试的成熟度问题也得到众多专家的关注，并提出了若干测试成熟度模型。

1.3.2　软件测试的国内外现状

在软件业发达的国家，尤其是美国，软件测试得到了软件从业人员的普遍重视，已经形成了一个独立的产业，发展程度较高。主要体现在如下方面：

(1) 针对软件测试的理论研究在高校和科研院所如火如荼地进行，不断有软件测试方面的学术会议召开，理论研究成果层出不穷，引导着测试产业的发展方向；很多高校还单独设置了软件测试专业或开设了测试课程，大批学生投身于软件测试的理论学习和实践。

(2) 在软件开发企业尤其是比较成功的企业如微软、IBM 等，软件测试被放在十分重要的位置上，公司设有独立的测试部门，配备专职的测试人员。例如，在微软公司，开发工程师与测试工程师的比例是 1∶2，而国内的一般公司是 6∶1。在开发一个软件的过程中，花费在测试上的人力、物力和资金比花费在编程上的多得多。很多大型开发项目，测试会占据项目周期一半以上的时间。以 IE 4.0 为例，代码开发时间为 6 个月，而稳定程序花费了 8 个月的时间。这些都为公司开发出高质量的软件提供了有力的保障。

(3) 涌现出许多著名的软件测试工具提供商，如 HP Mercury Interactive(MI)，Compuware，IBM Rational，Segue，Empirix 等。这些软件测试工具提供商的测试产品占据了全球的主流测试工具市场；同时，这些测试工具提供商所提供的测试方案和框架也极大地促进了软件测试技术的发展。

在我国，针对软件测试的理论研究起步于“六五”期间，在软件测试的实践方面则起步更晚。由于起步晚，我国无论在软件测试理论水平还是在测试产业方面，都与国际先进水平存在较大差距。但随着我国软件产业的高速发展及人们对软件质量的日益重视，软件测试正处于空前的快速发展时期，软件测试在我国正逐步形成一个独立的产业，主要体现在如下方面：

(1) 在我国，每两年召开一次的全国软件工程会议、全国容错计算会议都设有软件测试专题部分。从 2001 年起召开的全国测试学术会议，更是使软件测试得到了软件理论界前所未有的关注。

(2) 2003 年 10 月，国家人事部和信息产业部联合发文，决定将“软件评测师”资格考试纳入计算机技术与软件专业技术资格(水平)考试，使得软件测试人员的地位受到空前的重视。

(3) 国内著名的软件公司大都认识到软件测试的重要性，并成立了专职的软件测试队伍。虽然测试人员的数目和所占比例还不能与世界顶尖公司相比，但独立测试的意识在各公司得到了日益深化。

(4) “以测代评”正逐步成为我国科技项目择优支持的一项重要举措。如国家 863 计划对数据库管理系统、操作系统、办公软件、ERP 等项目的经费支持，都是通过第三方测试机构(即独立于软件开发商和用户的一方)科学、客观的测试结果来决定的。

(5) 各种软件测试职业培训机构如雨后春笋般涌现，软件测试还成为高校软件专业

普遍开设的课程，越来越多的人投身于软件测试的学习和实践中。

有理由相信，再经过若干年的发展，我国测试产业的发展水平会逐步接近国际先进水平，这也将促进我国软件产业的健康、快速发展。

但必须认识到，虽然近年来软件测试在国外和我国都取得了很大的进展，但其发展速度和水平仍难以适应高速发展的软件开发技术，软件测试从业人员面临着巨大的考验，主要体现在如下方面：

(1) 随着社会的发展，软件规模日益扩大，功能日益复杂，应用领域复杂多样，都对软件测试提出了新的考验。

(2) 面向对象开发是当前软件开发的主流技术，但面向对象软件的测试理论和技术还很不成熟。

(3) 对分布式系统的测试技术仍处于发展阶段。

(4) 对实时系统缺乏有效的测试手段。

(5) 对信息系统安全性的测试还处于起步阶段。

软件测试之所以发展得相对缓慢，主要有以下原因：

(1) 在软件测试的发展进程，尤其是软件测试的发展早期，软件开发人员对软件测试存在着敌对心理。认为软件测试一旦进行，就会找出程序中的许多错误，自己的劳动成果就会被部分否定，所以开发人员不愿意进行测试，更不愿意别人对自己的劳动成果进行测试。这种对软件测试的敌对心理，在很长一段时期内阻碍了软件测试的发展。

(2) 软件测试应贯穿软件开发的全过程，应把软件测试当作一个软件工程。也就是说，不能将软件测试当作是单一的几种方法或技能，软件测试也应遵循工程化原则，应当有完善的方法、过程和工具做支撑。但是，在软件概念出现以后的很多年，人们并没有软件工程的概念，这显然制约了软件测试的发展。20 世纪 70 年代以后，软件开发才进入软件工程阶段。软件工程的快速发展进一步加速了软件测试的发展。

(3) 软件测试是一门理论与实践高度结合的学科。没有系统的理论做支撑，软件测试就缺乏长足发展的后劲；而离开了测试实践，测试理论只能是纸上谈兵。但从现实情况来看，很少有人是既从事测试理论研究又参与大量测试实践的，这使得软件测试的发展面临着不平衡。

1.4 软件测试的发展趋势

纵观国内外软件测试现状，软件测试的主要发展趋势如下：

(1) 软件开发过程中的每个阶段都是可测试的，软件测试及早地介入软件开发的各阶段，如需求分析阶段和设计阶段。

(2) 软件测试成为一个完全独立的流程。测试可以贯穿软件开发的其他流程，并与之并发地执行。在某流程中，只要某测试达到准备就绪点，就可以进行该测试，而且测试是可迭代的。

(3) 面向对象的软件测试理论和技术不断发展。

(4) 分布式系统、实时系统的测试理论和技术不断发展。

(5) 软件测试人员的地位得到极大的提高。软件公司大都设置了独立的软件测试部门,配备有专职的软件测试人员,测试部门在公司各部门中有着重要的地位。软件测试不再被看作是一种破坏性工作,只有具备很强软件开发能力且性格良好的人才能胜任软件测试工作。

(6) 第三方测试迅速发展。第三方测试机构是一个具有中介性质的服务机构,它能通过自身专业化的测试技术和测试团队为客户和软件开发商提供有价值的服务。由独立的第三方软件测试机构完成测试,能够客观、公正地评价被测试软件,对软件开发商和用户来说都是有利的。

1.5　小结

软件测试是伴随着软件的发展而不断发展的。由于软件本身的特点,以及软件开发活动的特点,软件缺陷从来就难以根除,它可能给使用者造成或大或小的损失。

软件测试的目标是尽可能多地发现软件中的错误和缺陷,对软件质量进行度量和评估以提高软件质量,并检验软件系统是否满足用户需求。

对软件测试的看法中存在许多误区,只有认识这些误区才能更好地理解和实施软件测试。目前,软件测试在国内外均得到了快速发展,但软件测试的发展水平还远不能适应高速发展的软件开发技术,很多方面有待进一步研究和发展。

习　　题

1. 为何软件缺陷难以避免? 试谈谈你的观点。
2. 软件测试的目标是什么?
3. 测试和调试有何不同?
4. 为什么对软件进行穷举测试是不可能的?
5. 在一个软件的开发过程中,测试自动化的程度越高越好吗? 为什么?
6. 软件测试的发展落后于软件开发技术的发展水平,主要原因是什么?
7. 什么是第三方测试,其有何优点?
8. 你认为软件测试的发展趋势是什么?

第 2 章

软件测试基础

本章要点：

- 软件测试的定义。
- 软件测试的对象。
- 验证与确认的含义和区别。
- 软件测试的分类。
- 软件测试过程模型。
- 测试驱动开发的思想。
- 软件测试的原则。
- 软件测试文档的作用和分类。

通过第 1 章的学习，我们对软件测试有了一个大致的认识，但对其基本概念，如软件测试的定义、对象、分类等并不十分清楚，本章的目的在于介绍关于软件测试的基本概念、原则及软件测试文档，以及目前流行的测试驱动开发方法。这些内容是学习后续章节的基础。

2.1 软件测试概念

2.1.1 软件测试的定义和对象

1. 软件测试定义

1979 年，Glenford J. Myers 在其著作《软件测试的艺术》(*The Art of Software Testing*)中，对软件测试定义为："测试是为了发现错误而执行的一个程序或系统的过程。"这个定义不管是在早期还是当今，都有着相当大的影响。

20 世纪 80 年代早期，软件质量越来越多地得到人们的重视，软件测试的内涵正悄悄发生着改变。测试不再单纯地被认为是一个发现错误的过程，还被作为软件质量保证(Software Quality Assurance，SQA)(软件质量保证将在第 10 章进行介绍)的主要职能，包含软件质量评价的内容。

1983 年，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。"这个定义是对 Glenford J. Myers 定义的很好的补充，至今仍被引用。

1983 年，IEEE 提出的软件工程术语中对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。"这个定义明确地指出，软件测试的目的是检验软件系统是否满足需求。软件测试不再被认为是一个一次性的、只属于开发后期的活动，而应与软件的整个开发流程融为一体。

2. 软件测试的对象

早先许多人认为既然软件测试是希望发现程序中的错误，那么测试的对象就是程序。实际上，程序是软件开发过程中多个阶段工作的产物，如第 1 章所述，尽管软件中的错误会在程序中反映出来，但大部分错误是在编码之前造成的。

所以测试的对象不仅仅是程序，需求分析和设计工作也应列为测试的对象。甚至从一定意义上说，它们比程序更需要引起测试人员的注意。

2.1.2　验证与确认

目前，软件测试的效果难以尽如人意，是因为测试中尚存在如下问题：

(1) 测试工作多在软件开发的后期阶段进行，没有实现及早地、全面地进行测试，没有一个规范化、系统化的测试过程。

(2) 测试设计和测试操作未进行分离。

(3) 许多软件质量保证活动，如工作产品评估、可跟踪性分析、接口分析、关键性分析等是零散的、不自觉的行为，缺乏相应的规划。

为解决上述问题，人们提出了 V&V 概念，即验证(Verification)和确认(Validation)，这是软件测试领域十分有影响的概念。

1. 验证

验证即检验软件是否实现了预先定义的功能和其他特性，即判断软件开发每一阶段的活动是否已成功地完成，各开发阶段形成的软件配置是否保持一致。

2. 确认

确认也可理解为有效性确认。确认的目的在于判断交付使用的软件可否追溯到用户的需求。确认的作用是检验软件产品功能及其他特性的有效性。

3. 验证和确认的关系

从表面上看，对验证和确认的描述十分类似。它们的区别是什么？以下是 Boehm 对 V&V 两者的解释。

- Verification：Are we building the product right(我们在正确地构造软件吗)？
- Validation：Are we building the right product(我们在构造正确的软件吗)？

同样都是"正确"，但意义却不一样。举例来说，某软件的开发过程完全按照需求规格说明书和设计规格说明书的预期要求进行，每一开发阶段的工作都没有错误，且各阶段的软件配置具有一致性，显然此软件应该能通过验证，但一定能通过确认吗？不一定。因

为，也许该软件的需求分析和设计本身都存在缺陷，或者说开发人员（甚至也包括用户）原先对软件的功能及其实现的理解上存在缺陷，即使完全按预期的要求开发软件，也不一定能满足用户的最终真正需求。

所以，虽然定义中同样都含有“正确”，但确认定义中的“正确”的级别更高，它要求开发出来的软件对用户是真正有效的，能满足用户所有的最终需求，而这些需求中有些可能是潜在的，是用户先前都没有想到的。而验证定义中的“正确”只能说明软件开发的各阶段实现了既定的要求，但这些既定的要求本身可能存在问题。

验证和确认都属于测试活动。可以这样认为：

验证＋确认＝测试

验证和确认是不同级别的测试活动。

2.2 软件测试分类

下面从几个侧面介绍软件测试的分类。

2.2.1 按开发阶段分类

按开发阶段，软件测试可以划分为单元测试、集成测试、确认测试、系统测试和验收测试。

1. 单元测试

单元测试（Unit Testing）又称模块测试，是针对软件设计中的最小单位——程序模块进行正确性检验的测试。单元测试的目的在于发现程序模块内部可能存在的各种错误，检查各模块是否实现了详细设计说明中的模块功能、性能、接口及设计约束等方面的要求。单元测试应从程序模块的内部结构出发设计测试用例。多个模块可以并行地独立进行单元测试。

虽然单元测试是对某个模块的测试，但在测试某模块时，应考虑它与外界的联系。用一些辅助模块模拟与被测模块相联系的其他模块，从而达到测试被测模块的目的。

2. 集成测试

集成测试（Integrated Testing）也称为组装测试。在单元测试的基础上，将所有程序模块按照概要设计的要求组装成一个系统。集成测试的目的在于发现并排除在模块连接过程中可能出现的问题，最终构成符合概要设计要求的软件系统。

3. 确认测试

确认测试（Validation Testing）又称为有效性测试。确认测试的目的是检查已实现的软件系统是否满足需求规格说明书中规定的各种需求，以及软件配置是否完全、正确。

4. 系统测试

最终得到的软件不是一个孤立的对象，往往是作为整个目标系统的一个组成元素而存在的。

系统测试（System Testing）将通过确认测试的软件作为整个基于计算机系统的一个

元素，在实际运行环境下或模拟系统运行环境下，测试其与系统中其他元素（硬件、外设、网络、系统软件、支持平台等）能否正确地配置、连接，并满足用户需求。

系统测试的目的是通过与系统的需求定义比较，发现软件与系统定义不符合的地方。

5. 验收测试

验收测试（Acceptance Testing）即按项目任务书或合同、供需双方约定的验收依据文档对整个系统进行测试与评审，以决定是否接收软件系统。

验收测试是以用户为主的测试，但软件开发人员和 SQA（即 SQA 人员）也应参加。

2.2.2　按测试实施组织分类

按照实施测试的组织，可将测试分为 α 测试、β 测试和第三方测试。

1. α 测试

α 测试（Alpha Testing）属于开发方进行的测试，指软件开发方组织公司内部人员模拟各类用户对即将交付的软件产品（称为 α 版本）进行测试，以发现其中的错误并改正。α 测试的关键在于尽可能逼真地模拟软件的实际运行环境，并尽最大努力涵盖所有可能的用户操作方式。

α 测试的目的是评价软件产品的 FLURPS（Function，Localization，Usability，Reliability，Performance，Support），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。经过 α 测试调整的软件产品称为 β 版本。

2. β 测试

β 测试（Beta Testing）是用户进行的测试，但通常不等同于验收测试，即决定是否接收软件并不是 β 测试的目的。β 测试的目的在于帮助开发方在正式发布软件产品前对其进行最后的改进。

β 测试一般在 α 测试之后进行，是由大量用户在实际操作环境下对软件的 β 版本进行的测试。这些用户是与公司签订了支持软件产品预发行合同的外部客户，他们被要求以尽可能多的方式使用该软件，并将使用过程中出现的错误信息（真实的以及主观认定的）返回给开发方。开发方根据用户的错误报告，在正式发布软件产品之前对之进行一系列改进。

β 测试主要在于衡量产品的 FLURPS，着重于产品的支持性，包括文档、客户培训和支持产品生产能力。

3. 第三方测试

第三方测试是指由不同于开发方和用户方的组织进行的测试。通常模拟用户的真实操作环境，对软件进行确认测试。第三方测试有利于客观、公正地测试和评价软件。

2.2.3　按测试策略分类

根据测试实施策略的不同，软件测试可分为白盒测试、黑盒测试和灰盒测试。

1. 白盒测试

白盒测试（White-box Testing）又称为结构测试或逻辑驱动测试。顾名思义，“白盒”可理解为程序装在一个透明的盒子里，所以盒子内的程序对测试人员是可见的。

执行白盒测试的人员清楚地了解程序内部逻辑结构和处理过程，检查程序内部结构和路径是否达到了预期的设计要求。白盒测试方法将在第 4 章具体介绍。

2. 黑盒测试

黑盒测试(Black-box Testing)又称为功能测试或数据驱动测试。顾名思义，“黑盒”可理解为程序装在一个漆黑的盒子里，所以盒子内的程序对测试人员是不可见的。

执行黑盒测试的人员在已知软件应具有的功能的基础上，完全不考虑程序的内部逻辑结构和处理过程，在程序接口处进行测试，检查在需求规格说明书中规定的预期功能是否能正常实现。黑盒测试方法将在第 3 章具体介绍。

3. 灰盒测试

灰盒测试(Gray-box Testing)是一种介于白盒测试和黑盒测试之间的测试(故很多资料未对其单独介绍)。它基于程序运行的外部表现同时又结合程序内部逻辑结构来设计测试用例，执行程序并采集程序路径执行信息和外部用户接口结果。一般认为，集成测试阶段采用的测试策略近似于灰盒测试。

2.2.4 按测试执行方式分类

根据软件测试的执行方式，可将软件测试分为静态测试(Static Testing)和动态测试(Dynamic Testing)两种。

静态测试不实际执行程序，而是利用人工手段及静态测试工具完成对程序的静态测试。静态测试的主要目的是检查软件的表示与描述是否一致，没有冲突和歧义。静态测试将在第 4 章进行介绍。

动态测试则实际运行测试用例，以发现软件中的错误。

依据黑盒方法设计的测试是动态测试，白盒方法设计的测试则包括静态测试和动态测试两种类型，故对动态测试的介绍将贯穿于第 3 章和第 4 章。

2.2.5 其他测试方法和技术

在实际应用中，还有许多具体的测试类型，它们往往是为实现某特定目标而进行的测试。例如回归测试、迭代测试、功能测试、性能测试、安全性测试、可靠性测试、兼容性测试、可移植性测试、冒烟测试、用户界面测试、随机测试、引导测试、本地化测试等。

1. 回归测试

回归测试(Regression Testing)是为了验证对软件引入的修改的正确性及其影响而进行的测试。

在软件开发的任何一个阶段，只要软件发生了改变，就有可能引起一系列问题。软件的改变可能是因为发现了软件中的错误并做出了修改，也有可能是因为在集成或维护阶段加入了新的功能模块。

当软件中的错误被发现，往往需要修改的地方会有多处，包括表面的和深层次的。为了不遗漏任何需要修改之处，也为了避免修改对软件中未被修改之处造成的副作用，应进行回归测试；加入新的功能模块到软件中后，为使新功能能正常实现且不会对其他模块造成副作用，也应进行回归测试。

回归测试的工作量在整个软件测试过程中占有很大的比重，软件开发的各个阶段都会进行多次回归测试。

关于回归测试将在第 7 章详细介绍。

2. 迭代测试

迭代测试是从迭代的开发模式中延伸出来的。在迭代开发模式中，系统的开发是通过多次迭代完成的。每次迭代都包含需求分析、设计、编码、集成、测试等活动。每次迭代完成后，会出现新的迭代周期。通过一次次地迭代，系统增量式地集成若干新功能，直至最终实现系统应具备的全部功能。

在每个迭代周期中，测试工作由两方面组成：

(1) 对当前迭代周期产品的增量测试。

(2) 对原先迭代周期已完成功能的回归测试。

迭代开发模式继承了瀑布开发模式的优点：全面、计划性强和易于管理。更为重要的是，迭代开发模式将测试工作分布到每个迭代周期中，使测试工作提前进行，以尽早地发现软件中的缺陷，从而降低软件开发的风险和成本。

3. 功能测试

功能测试(Functional Testing)也称为行为测试(Behavioral Testing)，它根据产品特征、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们是否满足设计需求。一般把黑盒测试称为功能测试，当然这不是绝对的。功能测试有时也会用到白盒测试的方法，而黑盒测试有时可能是在对性能进行测试。

4. 性能测试

性能测试(Performance Testing)是评价一个产品或组件与性能需求是否符合的测试，包括负载(压力)测试、强度测试、容量测试、疲劳测试等类型。

性能测试是软件测试中的一个重点。对性能测试将在第 7 章进行具体介绍。

5. 安全性测试

安全性测试(Security Testing)的目的在于检测软件系统对非法侵入的防范能力。理论上讲，只要拥有足够多的时间和资源，任何系统都是可以侵入的，所以通过安全性测试及采取相应的措施后，应使非法侵入软件系统的代价大于被保护信息的价值，使非法侵入者得不偿失。

6. 可靠性测试

可靠性测试(Reliability Testing)的目的是测算在一定的环境下，系统能正常工作的概率。通常，用平均无故障时间(Mean Time Between Failures，MTBF)即两次失效之间的平均操作时间来衡量系统的可靠性。

7. 兼容性测试(配置测试)

兼容性测试(Compatibility Testing)有时也被称为配置测试(Configuration Testing)，但两者含义略有不同。一般说来，配置测试是为了保证软件在其相关的硬件上能够正常运行，而兼容性测试则主要是测试软件能否与其他不同的软件协作运行。

8. 可移植性测试

可移植性测试(Portability Testing)的目的在于验证软件能否被移植到指定的硬件或软件平台上。

9. 冒烟测试

冒烟测试(Smoke Testing)的对象是每一个新编译的需要正式测试的软件版本,目的是确认软件基本功能正常,可以进行后续的正式测试工作。冒烟测试的执行者是版本编译人员。

冒烟测试不能由测试小组独立建立,它应该是通过联合的方式,至少是在与开发人员达成一致的情况下建立的。冒烟测试的目标是显示其稳定性,而不是发现错误。必须在系统测试环境中进行冒烟测试。

冒烟测试的名称可以理解为该种测试耗时短,仅用一袋烟的功夫足够了。也有人认为是将其形象地类比于新电路板的基本功能检查(任何新电路板焊好后,应先通电检查,如果存在设计缺陷,电路板可能会短路,即出现冒烟的现象)。

10. 用户界面测试

用户界面测试(User Interface Testing)的目的在于测试用户界面的风格是否满足客户要求,包括用户友好性、人性化、易操作性等测试。

11. 随机测试

随机测试(Ad Hoc Testing)是没有书面测试用例的测试,主要是依据测试人员的经验对软件进行功能和性能抽查。

随机测试是根据测试文档执行用例测试的重要补充手段,是保证测试覆盖完整性的有效方式。

12. 引导测试

引导测试(Pilot Testing)是指在软件开发中验证系统在真实硬件和客户基础上处理典型操作的能力。在软件外包测试中,引导测试通常是客户检查软件测试公司测试能力的一种形式,只有通过了客户特定的引导测试,软件测试公司才能接受客户真实软件项目的软件测试。

13. 本地化测试

本地化测试(Localization Testing)的对象是软件的本地化版本,测试目的在于测试特定目标区域设置的软件本地化质量。从测试方法上可以分为基本功能测试、安装/卸载测试、当地区域的软硬件兼容性测试等。本地化测试的内容主要是软件本地化后的界面布局和软件翻译的语言质量,包含软件、文档和联机帮助等部分。

2.3 软件测试过程

随着测试技术的不断发展,人们更加关注测试的过程,对测试过程的管理已成为成功实施测试的重要保证。

软件测试过程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，同样地，测试过程的质量将直接影响测试实施的效果。软件测试过程和软件开发过程一样，都应遵循软件工程的原理。

2.3.1　软件测试过程模型

随着测试过程管理的发展，测试人员通过大量的实践总结出了很多很好的测试过程模型，如 V 模型、W 模型、H 模型等。这些模型将测试活动进行了抽象，并与开发活动进行了有机的结合，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 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 模型反映出了测试活动与分析设计活动的关系，如图 2-1 所示。图中，从左到右描述了基本的开发过程和测试行为，非常明确地标注了测试过程中存在的不同类型的测试，并且清楚地描述了这些测试阶段和开发过程中各阶段的对应关系。

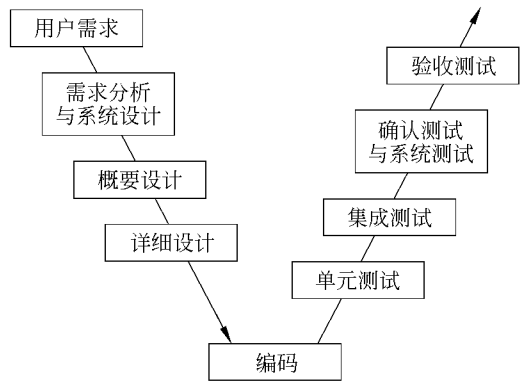

图 2-1　软件测试 V 模型

V 模型指出，单元和集成测试应检测程序的执行是否满足软件设计的要求；系统测试应检测系统功能、性能的质量特性是否达到系统要求的指标；确认测试和验收测试追溯软件需求规格说明书进行测试，确定软件的实现是否满足用户需要或合同的要求。

V 模型也存在一定的局限性。它仅仅把测试作为在编码之后的一个阶段，主要是针对程序进行的寻找错误的活动，对软件设计、需求分析等活动的测试要到后期才能完成。这样的测试顺序会使修复错误的代价大大增加，不利于提高软件开发和测试的效率。

2. W 模型

V 模型未能体现出“尽早地、全面地进行软件测试”的原则，为了弥补 V 模型的不足，W 模型出现了。

W 模型是由 Evolutif 公司提出的。相对于 V 模型，W 模型增加了软件各开发阶段中应同步进行的验证和确认活动。如图 2-2 所示，W 模型由两个 V 字型模型组成，分别表示测试和开发过程。从图 2-2 可以明显看出测试与开发的并行关系，也就是说，测试与开发是紧密结合的。

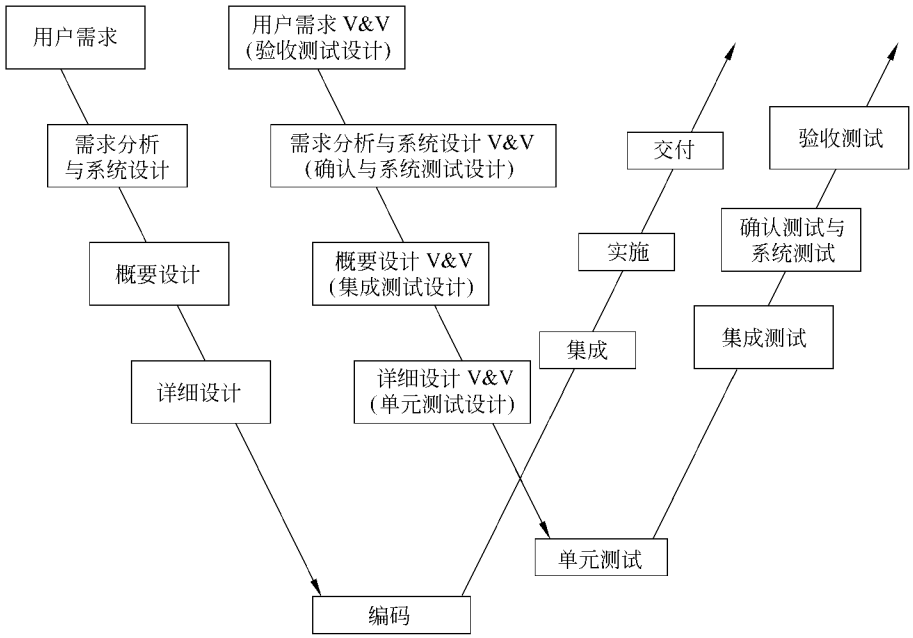

图 2-2 软件测试 W 模型

W 模型强调，测试伴随着软件开发的各阶段，测试的对象不仅仅是程序，需求分析、设计等同样需要测试。也就是说，测试与开发是同步进行的，当某一阶段的工作完成后，就可以进行测试。

W 模型有利于尽早地、全面地进行测试，以发现软件中存在的问题。例如，需求分析完成后，测试人员就应该参与到对需求的验证和确认活动中，以尽早地发现需求分析中存在的问题，并从可测试性角度为需求文档的编写提出建议。同时，测试人员结合前期对项目的把握，有利于及时了解项目的难度和测试中存在的风险，易于制定出完善的测试计划和方案，安排开发中各阶段的测试方法、进度和人员，使软件的开发过程进展顺利，提高软件测试和开发的效率。

W 模型也有利于全过程地测试。这是因为 W 模型中将测试与开发活动紧密结合起来，使测试人员充分关注开发过程，对开发过程的各种变更及时响应。例如，根据开发进度计划的变更及时调整测试进度和测试策略，以及依据需求的变更及时调整测试用例等。

W 模型也存在局限性。在 W 模型中，需求分析、设计、编码等活动被视为串行的，同时，测试和开发活动之间也是一种线性的关系，某开发活动完全结束后才可以正式开始进行测试，这样就无法支持迭代、自发性及变更调整。对于当前软件开发复杂多变的情况，W 模型并不能完全解决测试管理中面临的困惑。

3. H 模型

V 模型和 W 模型都存在不足之处，它们都把软件的开发过程中的需求分析、设计、编码等活动视为串行的。而大量的实践表明，各阶段保持严格的串行关系只是一种理想的状况，需求的变更等都会破坏这一理想状况，故与各开发阶段相对应的测试之间也不可能

保持严格的次序关系。同时，各层次的测试（单元测试、集成测试、系统测试等）也存在反复触发、迭代和增量关系。

为了解决以上问题，测试专家提出了 H 模型。它将测试活动完全独立出来，形成一个完全独立的流程，以将测试准备活动和测试执行活动清晰地体现出来，如图 2-3 所示。

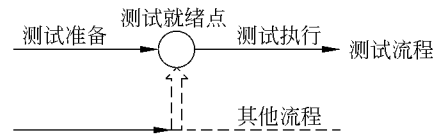

图 2-3　软件测试 H 模型

图 2-3 绘出的仅为整个软件生产周期中某个层次上的一次测试。图中标注的其他流程可以是任意的开发流程，例如设计流程或编码流程，也可以是非开发流程，如 SQA 流程，甚至是测试流程自身。只要测试准备活动完成，达到了测试就绪点，即可执行测试工作。

例如，在一个构件化 ERP 项目的系统测试过程中，由于前期需求难以确定，开发周期相对较长，为了进行更好的跟踪和管理，项目采用增量和迭代模型进行开发。整个项目开发共分三个阶段：第一阶段实现进销存的简单的功能和工作流；第二阶段实现固定资产管理、财务管理，并完善第一阶段的进销存功能；第三阶段增加办公自动化的管理。该项目每一阶段的工作是对上一阶段成果的一次迭代完善，同时叠加了新功能。

在该项目的系统测试过程中，根据 H 模型的思想，把系统测试作为一个独立的流程，达到相应的测试就绪点时即可执行测试。该系统的三个阶段相对独立，每一阶段完成的阶段产品亦具有相对独立性，可以作为系统测试的测试就绪点。故在该系统开发过程中，可开展三个阶段的系统测试，每个阶段系统测试具有不同的侧重点，以实现与各阶段开发的紧密结合，尽早发现软件中的错误，降低错误修复的成本。软件开发与系统测试过程的关系如图 2-4 所示。

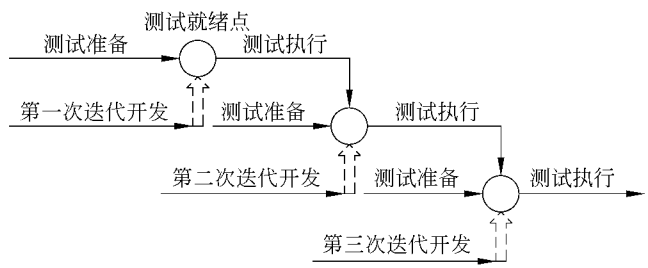

图 2-4　软件开发与系统测试的关系

H 模型使我们对软件测试有了更进一步的认识：软件测试不仅指测试的执行，还包括很多其他活动，如测试的准备；软件测试是一个独立的流程，可贯穿到软件产品整个生命周期中的任一流程，与之并发地进行；只要某个测试达到准备就绪点，测试执行活动就可以开展；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。

4. 其他模型

除上述几种常见模型外，业界还流传着其他模型，例如 X 模型、前置测试模型等。

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，再通过频繁的交接，通

过集成最终合成为可执行的程序。

前置测试模型体现了开发与测试的紧密结合，要求对每一个交付内容进行测试。使用该模型可以加快项目进度。

2.3.2 测试过程的实施策略

1. 测试过程模型选取策略

以上介绍的测试过程模型为人们提供了软件测试的流程和方法，为测试过程管理提供了重要依据，对指导开展测试工作具有极重要的意义。但任何测试模型都不是万能的，不可能有哪种模型完全适用于某项测试工作。应针对具体的开发项目灵活地选择测试过程模型，尽可能地利用各模型中对项目有实用价值的方面，不能为使用模型而使用模型。

一般说来，在实际的测试活动中，可以 W 模型为框架，及早全面地开展测试，同时灵活地运用 H 模型的独立测试思想，将测试与开发过程紧密结合，在达到恰当的测试就绪点时执行独立的测试工作，测试工作应是可迭代的。

2. 测试过程管理理念

测试过程管理理念是从无数的测试实践中提炼出来的，能指导测试人员成功地策划和开展测试过程，对于测试人员是不可或缺的精神财富。测试过程管理牵涉的范围非常广泛，包括过程定义、人力资源管理、风险管理等，以下仅介绍从测试过程模型中提炼出来的、对实际测试有指导意义的测试过程管理理念。

(1) 尽早测试

“尽早测试”是从 W 模型中抽象出来的理念。测试不应是在编码之后才开展的工作，测试与开发是两个相互依存的、并行的过程，在开发过程中的早期——需求分析阶段就应开展测试工作。

“尽早测试”主要包含两方面的含义，第一，测试人员早期参与到软件项目中，及时开展测试的准备工作，包括编写测试计划、制定测试方案以及准备测试用例。第二，尽早开展测试执行工作，一旦代码模块完成，就应该及时开展单元测试；一旦代码模块被集成为相对独立的子系统，便可以开展集成测试；一旦有产品提交，便可以开展系统测试工作。

尽早开展测试准备工作，使测试人员能够在早期了解测试的难度，预测测试的风险，有利于制定出完善的测试计划和方案，提高软件测试及开发的效率，规避测试中存在的风险。尽早开展测试执行工作，有利于测试人员尽早地发现软件中的缺陷，大大降低了错误修复的成本。

需要注意的是，“尽早测试”并不是盲目地提前测试活动，测试活动开展的前提是达到相应的测试就绪点。

(2) 全面测试

软件是程序、数据和文档的集合，因而软件测试不仅仅是对程序的测试，还应包括对软件副产品的全面测试，这是 W 模型中一个重要的理念。需求分析文档、设计文档作为软件的阶段性产品，直接影响到软件的质量。大量实践表明，软件中的大部分错误不是在编码阶段而是在编码之前的需求分析和设计中造成的。

“全面测试”主要包含下面两方面的含义：