第3章

软件测试基础

本章内容

- 软件测试的辩证观点
- 软件测试模型
- 软件测试过程
- 软件测试方法
- 软件测试用例

学习目标

(1) 了解软件测试的辩证观点。
(2) 掌握软件测试 W 模型。
(3) 掌握软件测试过程。
(4) 了解常用的软件测试方法。
(5) 掌握软件测试用例的关键要素。

3.1 认识软件测试

软件测试是软件工程中的一个重要环节,是贯穿整个软件开发生存周期的。软件测试主要的工作内容是**验证(verification)和确认(validation)**。

验证是检验开发出来的软件产品是否和需求规格及设计规格书一致,即是否满足软件厂商的生产要求。具体内容包括以下 3 点。

(1) 确定软件生存周期中的某一给定阶段的产品是否达到前阶段确立的需求的过程。

(2) 程序正确性的形式证明,即采用形式理论证明程序符合设计规约规定的过程。

(3) 评审、审查、测试、检查、审计等各类活动,或对某些项处理、服务或文件等是否与规定的需求相一致进行判断和提出报告。

确认就是检验产品功能的有效性,即是否满足用户的真正需求。确认包括静态确认和动态确认。

(1) 静态确认:不在计算机上实际执行程序,通过人工或程序分析来证明软件的正确性。

(2) 动态确认:通过执行程序,对执行结果做分析,测试程序的动态行为,以证实软件是否存在问题。

3.1.1　软件测试的辩证观点

G.J.Myers 认为“测试是为了证明程序有错，而不是证明程序无错误”。他的这一观点引出了对软件测试的争论：软件测试究竟是证明所有软件的功能特性是正确的，还是相反——对软件系统进行各种试探和攻击，找出软件系统中不正常或不工作的地方？这就是对于软件测试的正向思维和反向思维。

（1）正向思维：验证软件是“工作的”，针对软件系统的所有功能点，逐个验证其正确性。

（2）反向思维：证明软件是“不工作的”，不断思考开发人员理解的误区、不良习惯、程序代码边界、无效数据输入及系统的弱点，试图破坏系统、摧毁系统，目标就是发现系统中各种各样的问题。正如 G.J.Myers 强调的，一个成功的测试必须是发现缺陷的测试。

其实，这两个方面的认识都有一定道理，前者（证明或验证所有软件的功能特性是正确的）是从质量保证的角度来思考软件测试，后者（证明程序有错）从软件测试的目标和效率来思考软件测试，两者相辅相成。软件测试不仅是为了证明所有的功能都能正常工作，也是为了找出软件中那些不能正常工作、不一致性的问题。软件测试就是在这两者之间获得平衡，但对于不同的应用领域，两者的比重是不一样的。例如，国防、航天、银行等软件系统，承受不了系统的任何一次失效，因为任何失效都完全有可能导致灾难性的损失，所以强调前者，以保证非常高的软件质量。而一般的软件应用或服务，则可以强调后者，质量目标设置在“用户可接受水平”，以降低软件开发成本，加快软件发布速度，有利于市场的扩张。

3.1.2　软件测试的风险观点

一种观点认为，软件测试是“对软件系统中潜在的各种风险进行评估的活动”，这就引出软件测试的风险观点。软件测试自身的风险性是大家公认的，测试的覆盖率不能做到 100%；另一方面，软件测试的标准有时不清楚，软件规格说明书是测试中的一个标准，但也不是唯一的标准。因为规格说明书本身的内容完全有可能是错误的，它所定义的特性不是用户所需要的，所以，我们常常强调软件测试人员应该站在客户的角度去进行测试，除了发现程序中的错误，还要发现需求定义的错误、设计规格说明书的缺陷。但是，测试在大多数时间/情况下是由工程师完成的，而不是客户自己来做，所以又怎么能保证工程师和客户想

的一样呢?

对应软件测试的风险观点,产生了基于风险的测试策略:首先评估测试的风险,每个功能出问题的概率有多大?根据 Pareto 原则(也叫 80/20 原则),哪些功能是用户最常用的功能(约占 20%)?如果某个功能出问题,其对用户的影响又有多大?然后根据风险大小确定测试的优先级。优先级高的功能特性,测试优先得到执行。一般来讲,针对用户最常用的这 20%功能(优先级高)的测试会得到完全地、充分地执行,而低优先级功能的测试(用户不常用的功能,约占 80%)就可能由于时间或经费的限制,降低测试的要求、减少测试工作量,这样做风险并不是很大。

在"风险"观点的框架下,软件测试可以被看作一个动态的监控过程,对软件开发全过程进行检测,随时发现不健康的征兆,发现问题、报告问题,并重新评估新的风险,设置新的监控基准,不断地持续下去。这时,软件测试完全可以看作软件质量控制的过程。

3.1.3 软件测试的经济学观点

G.J.Myers 认为"一个好的测试用例在于它能发现至今未发现的错误",这体现了软件测试的经济学观点。实际上,软件测试经济学问题至今仍是业界关注的问题之一。经济学的核心就是要营利,营利的基础就是要有一个清楚的商业性目标,商业性目标是否正确,直接决定了企业是否营利。正如对软件质量的定义不仅局限于"和客户需求的一致性、适用性",而且要增加其他的要求——"开发成本控制在预算内、按时发布软件、系统易于维护"等,软件测试也一样,要尽快尽早地发现更多的缺陷,并督促和帮助开发人员修正缺陷。因为缺陷发现得越早,所付出的代价就越低,例如,在编程阶段发现一个需求定义上的错误,其代价将 10 倍于在需求阶段就发现该缺陷的代价。这就是从经济学的观点来说明测试进行得越早越好这样一个道理。

软件测试的对象是产品(包括阶段性产品,如市场需求说明书、产品规格说明书、技术设计文档、数据字典、程序包、用户文档等),而质量保证和管理的对象集中于软件开发的标准、流程和方法等上。

3.2 软件测试模型

软件测试和软件开发一样,都遵循软件工程、管理学原理。测试专家通过实践总结出了很多很好的测试模型。这些模型将测试活动进行了抽象,明确了测试与开发之间的关系,是测试管理的重要参考依据。

常见的软件测试模型有 V 模型、W 模型、H 模型、X 模型等。

3.2.1 V 模型

V 模型是由保罗·鲁克(Paul Rook)在 20 世纪 80 年代提出的,它是软件测试模型中最具有代表性的模型之一。V 模型是瀑布模型的变种,在瀑布模型的后半部分添加了测试工作,因整个开发过程构成一个 V 字形而得名,如图 3-1 所示。

V 模型从左往右依次是用户需求—需求分析与系统设计—概要设计—详细设计—编码—单元测试—集成测试—系统测试—验收测试。

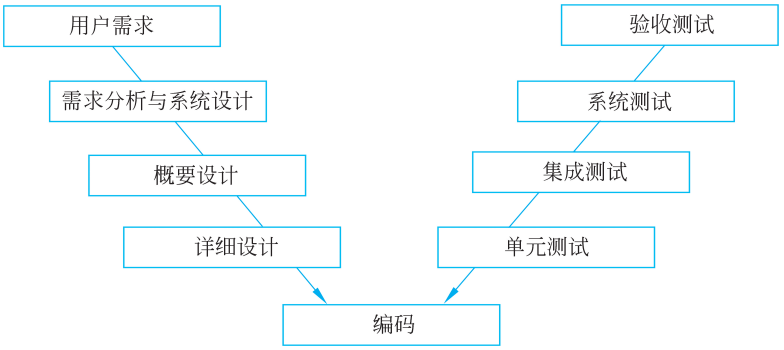

图 3-1　V 模型

（1）用户需求阶段：一般由甲方业务牵头部门成立需求编写小组，由小组人员编写并完善需求文档，有的项目也可能由乙方来完成。产出物为《××业务需求》《××新增功能工程业务需求》《××新增功能及扩容改造业务需求》等。

（2）需求分析与系统设计阶段：由甲方业务人员、用户、开发人员等完成，针对需求文档进行细致研讨和分析，产出物为《需求分析说明书》《需求规格说明书》等。

（3）概要设计阶段：该阶段由开发人员完成，产出物为《概要设计说明书》。

（4）详细设计阶段：该阶段由开发人员完成，产出物为《详细设计说明书》。

（5）编码阶段：该阶段由程序员完成，产出物为程序，即源代码。

（6）单元测试阶段：也叫模块测试，是最小的测试单位，理论上以白盒测试为主，测试对象一般是一个功能、类、函数、窗口、菜单等。实际工作中，考虑成本问题，一般由程序员自己完成。测试依据是《详细设计说明书》，从模型图中能看出，单元测试与详细设计相对应。

（7）集成测试阶段：也叫组装测试，组装过程一般是逐步完成的，所以会形成很多临时版本，理论上以黑盒测试为主，核心模块适当采用白盒测试。测试依据是《概要设计说明书》，从模型图中能看出，集成测试与概要设计相对应。

（8）系统测试阶段：是在所有功能组装完成后，对集成了硬件、软件、数据的完整系统进行的测试。测试的重点在于系统在真实环境下的正确运行以及系统的兼容性问题，测试方法为黑盒测试，测试依据是《需求规格说明书》《需求文档》等文档。从模型图中能看出，系统测试与需求分析相对应。

（9）验收测试阶段：也叫用户接受度测试，是由用户参与的验收过程，包括 alpha 测试和 beta 测试。从模型图中能看出，验收测试阶段与用户需求阶段相对应。

V 模型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，描述了这些测试阶段和开发过程期间各阶段的对应关系。V 模型的这些特性既有优点也有其局限性。

（1）V 模型的优点：V 模型强调软件开发的协作和速度，将软件实现和验证有机地结合起来，在保证较高的软件质量情况下缩短开发周期。V 模型适合工程量小、人力资源少并且开发过程中改动不大的项目。

（2）V 模型的局限性：V 模型仅把测试过程放在需求分析、系统设计、编码之后的一个阶段，忽视了测试对于需求的分析和验证。我们对需求的验证，对系统设计的验证，到后期的验收测试才有可能被发现，对于测试需要尽早进行的原则在 V 模型中没有体现，这是 V

模型的局限。


W 模型

3.2.2 W 模型

W 模型又称双 V 模型,它由 V 模型演变而来,弥补了 V 模型的不足。左边的 V 是开发的生命周期,右边的 V 是测试的生命周期,如图 3-2 所示。W 模型强调测试伴随着整个软件开发周期,测试的对象不仅是程序,需求、功能和设计同样需要测试。W 模型的这些特性既有优点也有其局限性。

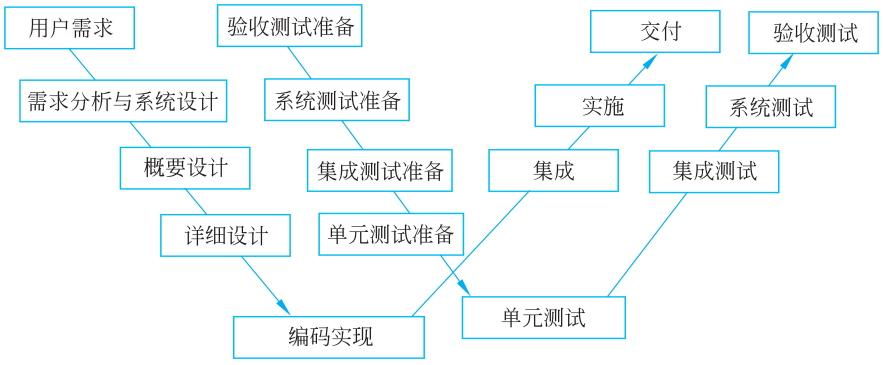

图 3-2 W 模型

(1) W 模型的优点:W 模型是一个开发与测试并行的模型,体现了尽早测试和不断测试原则,有利于及时地了解项目的测试风险,来及早地执行相应的应对方案,加快项目的进度。同时,W 模型强调了测试计划等工作的先行以及对系统需求和系统设计的测试,有利于尽早地全面地发现问题。

(2) W 模型的局限性:W 模型中,需求、设计、编码仍然是串行进行的,测试和开发保持线性的关系,上一个阶段完成之后才能进行下一个阶段,不能够很好地支持迭代的开发模型。

3.2.3 H 模型

为了解决 V 模型与 W 模型存在的问题,有专家提出了 H 模型。H 模型将测试活动完全独立了出来,形成一个完全独立的流程,这个流程将测试准备活动和测试执行活动清晰地体现出来。在 H 模型中,测试流程和其他工作流程是并发执行的,只要某一个工作流程的条件成熟就可以开始进行测试,其过程如图 3-3 所示。

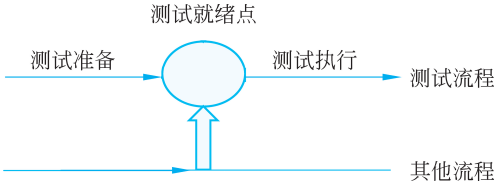

图 3-3 H 模型

在 H 模型中测试级别不存在严格的次序关系,软件生命周期的各阶段的测试工作可以反复触发、迭代,即不同的测试可以反复迭代地进行。例如,在概要设计工作流程上完成一

个测试，只是体现了软件生命周期中概要设计层次上的一个测试“微循环”。在实际测试工作中，H 模型并无太多指导意义，读者重点是理解其中的设计意义。H 模型的这些特性既有优点也有其局限性。

1. H 模型的优点

软件测试完全独立，贯穿整个生命周期，且与其他流程并发进行；软件测试活动可以尽早准备、尽早执行，具有很强的灵活性；软件测试可以根据被测物的不同而分层次、分阶段、分次序执行，同时也是可以被迭代的；H 模型体现了尽早测试和不断测试原则。

2. H 模型的局限性

管理要求高：由于模型很灵活，必须要定义清晰的规则和管理制度，否则测试过程将非常难以管理和控制。

技能要求高：H 模型要求能够很好地定义每个迭代的规模，不能太大也不能太小。

测试就绪点分析困难：很多时候，很难确定测试准备到什么时候是合适的，就绪点在哪里，就绪点的标准是什么，这就对后续的测试执行启动带来很大困难。

人员要求高：对于整个项目组的人员要求非常高，只有在很好的规范制度下，大家都能高效地工作，才能较好地完成测试。例如，你分了一个小的迭代，但是因为人员技能不足，使其无法有效完成，那么整个项目就会受到很大的干扰。

事实上，随着软件质量要求越来越为人们所重视，软件测试也逐步发展成为一个独立于软件开发的一系列活动，就每一个软件测试的细节而言，它都有一个独立的操作流程。例如，现在的第三方测试，就包含从测试计划和测试用例编写，到测试实施以及测试报告编写的全过程，这个过程在 H 模型中得到了相应的体现，表现为测试是独立的。也就是说，只要测试前提具备了，就可以开始进行测试了。

3.2.4　X 模型

X 模型的设计原理是将程序分成多个片段反复迭代测试，然后将多个片段集成再进行迭代测试，如图 3-4 所示。

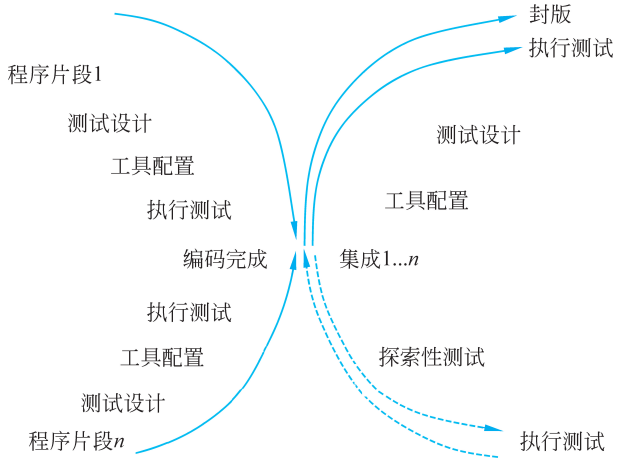

图 3-4　X 模型

X 模型左边是单元测试和单元模型之间的集成测试，右边是功能的集成测试，通过不断

的集成最后成为一个系统,如果整个系统测试没有问题就可以封版发布。X 模型并不要求在进行作为创建可执行程序(图中右上方)的一个组成部分的集成测试之前,对每一个程序片段都进行单元测试(图中左侧的行为)。从图 3-4 中可以看到,X 模型还定位了探索式测试,探索式测试是不进行事先计划的特殊类型的测试,能够帮助测试人员在测试计划之外发现更多的错误。

X 模型呈现了一种不断迭代的动态测试过程,更符合企业实际情况。图中多根并行的曲线表示变更可以在各个部分发生。X 模型的这些特性既有优点也有其局限性。

(1) X 模型的优点:X 模型可以很好地处理测试与开发的交接过程(交接的过程是一个时间段,而不是一个点);公司可以根据自身的情况确定是否要做单元测试,还是直接做系统测试;测试应该是一个不断迭代的过程,直到封版发布;提倡探索性测试。

(2) X 模型的局限性:探索性测试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。

软件测试模型对指导测试工作的进行具有重要的意义,但任何模型都不是完美的。上面共介绍了 4 种软件测试模型,在实际测试工作中,测试人员更多的是结合 W 模型与 H 模型进行工作,软件各个方面的测试内容是以 W 模型为准,而测试周期、测试计划和进度是以 H 模型为指导。X 模型更多的是作为最终测试、熟练性测试的模板,例如,对一个业务测试已经有两年时间,则可以使用 X 模型进行模块化的、探索性的方向测试。

3.3 软件测试过程

随着软件测试技术的发展,测试工作由原来单一的寻找缺陷逐渐发展成为预防缺陷、探索测试、破坏程序的过程,测试活动贯穿于整个软件生命周期中,故称为全程软件测试。全程软件测试,强调整个软件生命周期中各阶段的测试活动。无论是需求阶段、开发阶段,还是测试阶段,都需要确定在当前阶段测试活动的内容,确保每个阶段的质量,才能保证软件产品最终的质量。

全程软件测试的测试活动贯穿软件开发的整个生命周期,各个阶段测试活动内容包括:

1. 测试需求分析

测试需求分析是整个测试活动中除了测试用例设计之外最重要的部分。测试需求分析的目的是理解需求,理解业务,弄清楚待测产品有哪些功能需求,有哪些非功能性需求;明白待测产品的用户群体是什么,用户会如何使用待测产品。

2. 测试计划制定

当对需求有完整和全面的理解后,接下来需要制定详细的测试计划,为即将开始的测试工作做好充足的准备。测试计划包括以下内容。

- 资源估算:整个项目需要多少资源?硬件资源,人力、时间资源等。
- 进度控制:每个测试活动时间点控制。
- 风险控制:对于在测试活动过程中出现问题的解决方案。
- 资源配置:如何更有效率地使用资源。
- 验收标准:文档、项目、测试过程的验收标准定义。
- 测试策略:测试中使用的测试策略。

3. 测试用例设计

测试用例设计是软件测试工作的灵魂。任何一项测试活动的核心都是测试思维，即如何进行测试，测试用例就是测试思维的体现。功能的测试优先级、如何操作、输入什么数据、应该有什么的结果等都体现在测试用例中。

4. 测试用例评审

测试用例的评审无疑是为了给测试用例进行查漏补缺。测试用例的评审一般包括测试内部评审和项目组评审两种形式。项目组评审要求项目相关人员（开发、测试、产品）参与评审，一般是会议的形式。由于测试用例的数量关系，会议上评审会占用很长的时间，可能造成时间资源的浪费。

5. 测试执行

依据不同迭代版本所完成的功能执行测试用例，对无法通过的测试用例，确定复现步骤后或以添加截图的形式向开发提交 bug。前面的工作做得充足的话，在测试执行的时候就会非常简单了。但是在执行的过程中各部门的协作、沟通以及各项文档的输出却很复杂。例如，测试人员与开发人员的日常沟通：

“×××，我这有个问题，你过来看下。”

“什么问题？你演示下我看看。”“这不是问题，这个地方只能这样做。”或者“这不是问题，我刚刚跟需求确认过的。”

“这样做不合逻辑啊！”

“那你说怎么处理？”

“我觉得应该……处理。”

“你先跟需求确认下吧。”

要提高测试人员与开发人员的沟通效率，需要整个项目组有意识地培养健全的工作流程，建立完善的需求变更体系，流程上控制需求变更。同时，要求测试人员一开始提问题的时候就把问题的特征、位置、操作步骤、截图都一目了然地提交给开发人员，这样可以减少测试人员与开发人员的沟通成本。

6. 缺陷管理

在开发阶段，测试人员最重要的产出就是软件缺陷。要体现软件缺陷的价值，应该关注缺陷的质量、缺陷的管理以及缺陷分析。

缺陷管理是软件测试活动中极其重要的一环，很多时候测试用例并没有发现多少缺陷，反而在运行程序的过程中发现了很多缺陷，这些缺陷就是对测试用例的补充，对之后的测试可以提供思路。

7. 测试报告

完成测试后，提交测试报告，给出此次测试过程中的数据，例如，测试用例的数量、发现缺陷的总数、各个严重程度的缺陷数量、总共修复的缺陷数量以及缺陷修复率等。每一次软件通过测试准则发布后，测试人员都应该持续反馈、改进、总结每个发布版本中遇到的问题，从问题中总结经验，提高整个软件生命周期的质量。

全程软件测试，关注的是在整个软件生命周期中各个阶段的测试活动，通过对各个阶段的过程质量把控，最终提高项目团队的综合能力，从而提高软件产品的测试质量。软件产品的质量不是测试决定的，而是整个项目团队决定的，是在整个项目构建过程中，通过一次次

的优化过程,不断地总结成长来提高的。


软件测试方法

3.4 软件测试方法

软件测试是软件开发过程的重要组成部分,是用来确认一个程序的品质或性能是否符合开发之前所提出的要求。软件测试的目的,首先是确认软件的质量,其一方面是确认软件做了所期望的事情(Do the right thing),另一方面是确认软件以正确的方式来做了所期望的事情(Do it right)。软件测试的第二个目的是提供信息,例如,提供给开发人员或程序经理的反馈信息,为风险评估所准备的信息等。软件测试的第三个目的是测试软件开发的过程。如果一个软件产品开发完成之后发现了很多问题,说明此软件的开发过程很可能是有缺陷的,因此软件测试要保证整个软件开发过程是高质量的。

常用的软件测试方法有黑盒测试、白盒测试、基于风险的测试、基于模型的测试等。

3.4.1 黑盒测试

黑盒测试,顾名思义就是将被测系统看成一个黑盒,从外界取得输入,然后再输出,完全不考虑程序内部结构和处理过程,是从软件的外部表现来发现被测系统的缺陷和错误。黑盒测试基于需求文档,在程序界面处进行测试,检查程序是否按照需求规格说明书的规定正常实现。黑盒测试又称功能测试、数据驱动测试、基于规格说明的测试。

黑盒测试的优点如下。

(1) 比较简单,不需要了解程序内部的代码及实现。

(2) 与软件的内部实现无关。

(3) 从用户角度出发,能很容易地知道用户会用到哪些功能,会遇到哪些问题。

(4) 基于软件开发文档,因此能清楚地了解软件实现了文档中的哪些功能。

(5) 在做软件自动化测试时较为方便。

黑盒测试的局限性如下。

(1) 不可能覆盖所有的代码,覆盖率较低,大概只能达到总代码量的 30%。

(2) 自动化测试的复用性较低。

3.4.2 白盒测试

白盒测试通过对程序内部结构的分析、检测来寻找问题。白盒测试可以把程序看成装在一个透明的白盒子里,也就是清楚了解程序结构和处理过程,检查是否所有的结构及路径都是正确的,检查软件内部动作是否按照设计说明的规定正常进行。因此,白盒测试需要知道程序内部的设计结构及具体的代码实现,并以此为基础来设计测试用例。

白盒测试的优点是能帮助软件测试人员增大代码的覆盖率,提高代码的质量,发现代码中隐藏的问题。

白盒测试的局限性如下。

(1) 程序运行会有很多不同的路径,不可能测试所有的运行路径。

(2) 白盒测试是基于代码的测试,只能测试开发人员做得正确与否,而不能测试系统设计的正确与否,因此可能会漏掉一些功能需求。

（3）系统庞大时，白盒测试开销会非常大。

3.4.3　基于风险的测试

基于风险的测试是指评估测试项的优先级，先做高优先级的测试项，如果时间或精力不够，低优先级的测试项可以暂时先不做。基于风险的测试通常根据一个软件的特点来确定：如果一个功能出了问题，它对整个产品的影响有多大，这个功能出问题的概率有多大？如果出问题的概率很大，出了问题对整个产品的影响也很大，那么在测试时就一定要覆盖到。对于一个用户很少用到的功能，出问题的概率很小，就算出了问题影响也不是很大，那么如果时间比较紧，就可以考虑不测试。

基于风险测试的两个决定因素就是：该功能出问题对用户的影响有多大，出问题的概率有多大。其他一些影响因素，如复杂性、可用性、依赖性、可修改性等也会影响基于风险测试的测试顺序，测试人员主要根据事情的轻重缓急来决定测试工作的重点。

3.4.4　基于模型的测试

模型是系统的抽象，实际上就是用语言把一个系统的行为描述出来，定义出它可能的各种状态，以及它们之间的转换关系，即状态转换图。基于模型的测试是利用模型来生成相应的测试用例，然后根据实际结果和原先预想的结果的差异来测试系统。

3.5　软件测试用例


测试用例

测试用例（Test Case）是为特定的目的而设计的一组测试输入、执行条件和预期结果的文档，其作用是测试是否满足某个特定需求。测试用例是指导测试工作进行的依据。

1. 测试用例的组成

标准的测试用例通常由以下几个要素组成。

（1）用例编号：测试用例的唯一标识。

（2）模块：标明被测需求具体属于哪个模块，主要是为了更好地识别以及维护用例。

（3）用例标题：又称为测试点，就是用一句话来描述测试用例的关注点。每一条用例对应一个测试目的。

（4）优先级：根据需求的优先级别来定义。高优先级要覆盖核心业务、重要特性以及使用频率比较高的部分。

（5）前提条件：用例在执行之前需要满足的一些条件，否则测试用例无法执行。例如，一些测试环境，或者需要提前执行的操作。

（6）测试数据：在执行测试时，需要输入一些外部数据来完成测试。这些数据根据测试用例的统计情况来确定，有参数、文件或者数据库记录等。

（7）测试步骤：测试用例的步骤描述，执行人员可以根据测试步骤完成测试的执行。

（8）期望结果：是测试用例中最重要的部分，主要用来判断被测对象是否正常。要根据需求来描述用户的期望。

（9）实际结果：PASS 通过，FAIL 失败，N/A 未执行。

2. 测试用例的优先级

标准的测试用例通常都包含优先级这一要素,用来体现用例的功能的重要程度。在划定用例的优先级时,没有什么严格的标准和界限,通常会根据系统需求划分成以下四个等级。

(1) P0:核心功能测试用例(冒烟测试),确定此版本是否可测的测试用例。此部分测试用例如果失败,其他测试用例就可以不用执行了。

(2) P1:高优先级测试用例,最常执行,用来保证功能是稳定的。包含基本功能测试和重要的错误、边界测试。

(3) P2:中优先级测试用例,更全面地验证功能的各个方面,包含异常测试、边界、中断、网络、容错、UI 等测试用例。

(4) P3:低优先级测试用例,不常被执行,一般包含性能、压力、兼容性、安全、可用性等。

3. 测试用例的特性

测试用例通常的主要特性如下。

(1) 有效性:测试用例能够被使用,且不同人员得到的测试结果一致。

(2) 可复用性:可以重复使用(回归测试)。

(3) 易组织性:可以分门别类地提供给测试人员参考和使用。

(4) 可评估性:从测试管理的角度,测试用例的通过率和软件缺陷的数目是软件产品质量好坏的测试标准。

(5) 可管理性:测试人员工作量的计算和绩效考核。

知识拓展: 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的一种测试方法。

4. 测试用例的作用

测试用例通常使得测试工作可重复,是测试执行者工作的重要依据,其作用主要体现在以下几个方面。

(1) 帮助测试员理清思路,避免遗漏。测试用例提前准备好的测试数据可以帮助测试员做到心中有数,不会一个测试点重复测试好多次,也不会漏掉测试点。

(2) 测试用例的执行结果是评估测试结果的度量基准。如果设计全面覆盖需求的用例,测试用例都执行通过,发现的问题全部修改,即可放心交付给客户使用。

(3) 测试用例是分析缺陷的标准。因为测试用例中会详细描述期望结果,这个期望结果其实就是分析是不是有缺陷的一个标准。和预期结果一致的,就是没有缺陷;反之,和预期结果不一致,就是存在缺陷,需要进行修复。

5. 典型的测试用例表

一份好的用例可以帮助提高测试效率;相反,一份不好的用例,会给软件测试员的工作带来很多隐患,降低测试效率。表 3-1 是实际测试工作中使用的典型测试用例表,表 3-2 是本书中为了表达的简洁性使用的简化版测试用例表。

表 3-1 典型的测试用例表

测试用例 ID	测试用例的 ID(由案例管理系统自动生成,方便跟踪管理)
测试用例名称	

续表

产品名称		产品版本	
功能模块名称		测试平台	
用例入库者		用例更新者	
用例入库时间		用例更新时间	
测试功能点	测试的功能检查点		
测试目的	该测试案例的测试目的		
测试级别	测试级别：主路径测试、烟雾测试、基本功能测试、详细功能测试		
测试类型	测试类型：功能测试、边界测试、异常测试、性能测试、压力测试、兼容测试、安全测试、恢复测试、安装测试、界面测试、启动/停止测试、文档测试、配置测试、可靠性测试、易用性测试、多语言测试		
预置条件	对测试的特殊条件或配置进行说明		
测试步骤	详细描述测试过程，案例的操作步骤建议少于 15 个		
预期结果	预期的测试结果		
实际结果	实际的测试结果		

表 3-2　本书中简化版的测试用例表

用例编号	输　入			预期输出	备　注
	地区码	前　缀	后　缀		
1					
2					

3.6　软件测试的现状

我国的软件测试技术研究起步于“六五”期间，主要是随着软件工程的研究而逐步发展起来的。以前，软件测试一直被中小 IT 企业所忽视，只有一些知名企业才有专门的软件测试人员。软件产业发展到今天，更多的国内企业开始认识到测试的重要性。根据中国调研报告网发布的《2019 年中国软件测试行业现状研究分析与市场前景预测报告》显示，软件测试企业以非外包公司为主，其中，传统 IT 企业、互联网企业数量占比超过 50%。目前，软件测试企业对软件测试已有较高的认可度和重视度。随着企业对软件测试的重视度不断提升，企业测试人员与开发人员比基本保持在 1∶3 的比例，比例在 1∶7 以上的近几年来下降趋势明显。

目前，软件测试正在逐步成为一个新兴产业，主要体现在以下几个方面。

1. 软件测试的重要性和规范性不断提高

国家各部委、各行业正在通过测试来规范软件行业的健康发展，通过测试把不符合行业标准的软件挡在门外，对行业信息化的健康发展起到了很好的促进作用。在信息产业部关

于计算机系统集成资质以及信息系统工程监理资质的认证中,软件测试能力已经被定为评价公司技术能力的一项重要指标。

2. 从手工向自动化测试方式的转变

传统的项目测试还是以手工为主,测试人员根据需求规格说明书的要求,与测试对象进行“人机对话”。大量的手工工作增加了项目人力成本和沟通成本,造成低效率以及高差错率。针对企业的网络应用环境需要支持大量用户和复杂的软硬件应用环境,手工测试的工作量也越来越大,自动化测试及管理已经成为项目测试的一大趋势。

自动化测试通过测试工具和其他手段,按照测试工程师的预定计划对软件产品进行自动的测试,它能够完成许多手工无法完成或者难以实现的测试工作,更好地利用资源,将烦琐的任务赋予自动化方式,从而提高测试准确性和测试人员的积极性。正确、合理地实施自动化测试,能够快速、全民地对软件进行测试,从而提高软件质量、节省经费,缩短产品发布周期。

3. 测试人员需求逐步增大,素质不断提高

随着 IT 业的迅猛发展,软件外包服务已成为继互联网和网络游戏后的第五次全球浪潮。由于外包对软件质量要求很高,国内软件企业要想在国际市场上立足,就必须重视软件质量,而作为软件质量的把关者,软件测试工程师日渐“走俏”。目前在国内 120 万软件从业人员中,真正能担当软件测试职位的不超过 5 万,而目前高等教育中专业的软件测试教育近于空白,独立开设软件测试课程的高校非常少,这就形成测试人才紧缺、需求不断增大的现象。据分析,目前国内软件测试的人才需求缺口超过 20 万人。

为了在软件市场保持竞争力,软件企业开始加强和重视测试人员的选拔、培养和知识培训。一方面,对测试人员的素质和要求逐步提高,测试人员不仅应掌握相关计算机知识背景、软件工程基本知识,熟悉项目编程语言、熟悉项目技术架构及需求内容,而且要求工作有责任感、独立分析能力及团队精神等方面;另一方面,软件企业为测试人员提供了进一步的培训机会,以应对各种测试项目的复杂情况。

4. 测试服务体系初步形成

随着用户对软件质量的要求越来越高,信息系统验收不再走过场,而要通过第三方测试机构的严格测试来判定。“以测代评”正在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、办公软件等项目的经费支持,都是通过第三方测试机构科学客观的测试结果来决定。随着第三方测试机构的蓬勃发展,在全国各地新成立的软件测试机构达十多家,测试服务体系已经基本确立起来。

2018 年以后,软件测试岗位发展相对稳定,未来软件测试行业的前景体现在以下两个方面。

(1) 软件测试在未来 5～10 年内发展会很快,人口缺口很大。软件企业要靠产品去占领市场,研发部门要把产品开发出来,需要软件开发部门和软件测试部门的合作才能完成,开发部门开发的软件符不符合客户的需求,能不能实现所需诉求,都需要测试人员去保障。测试人员可谓是一个软件企业生存的命脉,测试这关过不了,做出来的产品将会漏洞无数。

(2) 软件测试行业对测试人员的技能要求也越来越高。以前很多测试人员由于知识不成体系,技术掌握不牢固,只能应对一些简单的测试工作。但是随着软件行业的发展,企业更多需要的是一些技术层级稍微高一点的软件测试人才。

随着软件产业的发展，软件产品的质量控制与质量管理正逐渐成为软件企业生存与发展的核心。几乎每个大中型 IT 企业的软件产品在发布前都需要大量的质量控制、测试和文档工作，而这些工作必须依靠拥有娴熟技术的专业软件人才来完成。软件测试工程师就是这样的一个企业重头角色。业内人士分析，软件测试工程师职位的需求主要集中在沿海发达城市，其中，北京和上海的需求量分别占去 33％和 29％。民企需求量最大，占 19％，外商独资欧美类企业需求排列第二，占 15％。

习　　题

一、判断题

1. 单元测试通常由开发人员进行。（　　）
2. 在任何情况下执行回归测试时，都应将以前测试过的用例全部执行一遍。（　　）
3. 穷尽测试是不可能的。（　　）
4. 测试旨在防止错误的发生。（　　）
5. 测试工作具有创造性。（　　）
6. 测试用例应由测试输入数据和对应的实际输出结果这两部分组成。（　　）
7. 黑盒测试又称数据驱动测试。（　　）
8. 黑盒测试又称基于规格说明的测试。（　　）
9. 白盒测试又称逻辑驱动测试。（　　）
10. 白盒测试又称基于程序的测试。（　　）
11. 白盒测试又称结构测试。（　　）

二、选择题

1. 软件测试越早开始越好，因此软件测试应从（　　）阶段开始。
   A. 需求分析（编制产品说明书）
   B. 设计
   C. 编码
   D. 产品发布
2. 软件测试类型按开发阶段划分为（　　）。
   A. 需求测试、单元测试、集成测试
   B. 单元测试、集成测试、系统测试、验收测试
   C. 单元测试、集成测试、确认测试
   D. 调试、单元测试、功能测试
3. 软件测试的目的是（　　）。
   A. 发现程序中的所有错误
   B. 尽可能多地发现程序中的错误
   C. 证明程序是正确的
   D. 调试程序
4. 下列不属于测试用例设计原则的是（　　）。
   A. 测试用例应具有代表性　　B. 测试用例的测试结果具有可判定性

C. 测试用例的测试结果具有可再现性　　D. 测试用例的测试结果不具有可再现性

5. 以下(　　)不属于黑盒测试方法的优势。

A. 比较简单,不需要了解程序内部的代码及实现

B. 与软件的内部实现无关

C. 从用户的角度出发,能很容易知道用户会用到哪些功能

D. 不可能覆盖所有的代码,覆盖率较低

6. 关于软件测试,以下说法错误的是(　　)。

A. 测试能提高软件的质量,但是提高质量不能依赖测试

B. 测试只能证明缺陷存在,不能证明缺陷不存在

C. 开发人员测试自己的程序后,可作为该程序已经通过测试的依据

D. 80%的缺陷聚集在20%的模块中,经常出错的模块改错后还会经常出现

7. 软件测试是采用(　　)执行软件的活动。

A. 测试用例　　B. 输入数据　　C. 测试环境　　D. 输入条件

三、简答题

简述什么是测试用例。