第5章 单元测试、集成测试和系统测试

5

【本章学习目标】

- 了解单元测试、集成测试、系统测试的基本概念。
- 掌握单元测试、集成测试及系统测试的测试内容、方法和过程。
- 掌握测试报告的撰写。

本章首先介绍单元测试的基本概念、测试内容、测试方法和过程，再介绍集成测试的基本概念、测试内容、测试方法和测试过程，最后介绍系统测试的主要内容、测试方法和测试过程。

5.1 单元测试基本概念

在第2章中，已经介绍了软件开发与软件测试的关系。由于软件开发采用工程方法，将大的系统划分为小的模块进行设计、开发和实现，就是模块化方法。对这些模块进行测试，就是单元测试。

单元测试又称模块测试，是对已实现软件的最小单元进行测试，发现其中存在的软件缺陷，以保证构成软件的各个单元的质量。这些最小单元可以是一个类、一个函数或一个子程序。

5.1.1 单元测试的任务

单元测试检查每个模块是否能正确实现详细设计说明书中的功能、性能、接口和其他设计约束要求，确保每个单元都能被正确地编码。单元测试要检查如下问题。

(1) 该单元能否完成其特定的功能和性能。

(2) 该单元的运行能否满足特定的逻辑覆盖。

(3) 在运行该单元时，其内部的数据能否保持完整性。全局变量的处理，内部数据的形

式、内容及相互关系等不应出现错误。

(4) 是否能处理符合要求和不符合要求的数据,在数据边界条件上,能否正常运行。

(5) 对该单元中发生的错误,是否采取有效的处理措施。

综合上面几个方面,除了对各模块的功能与性能测试外,单元测试主要是对各个模块的5个基本特性进行评价。

1. 模块的接口

模块的接口与参数有关,主要检查如下内容。

(1) 实际参数与形式参数的个数是否相等。

(2) 实际参数与形式参数的属性是否匹配。

(3) 实际参数与形式参数的单位是否匹配。

(4) 调用其他模块时所给实际参数的个数是否与被调模块的形式参数个数相等。

(5) 调用其他模块时所给实际参数的属性是否与被调模块的形式参数属性匹配。

(6) 调用其他模块时所给实际参数的单位是否与被调模块的形式参数单位匹配。

(7) 调用内部函数所用参数的个数、属性和次序是否正确。

(8) 是否存在与当前入口点无关的参数引用。

(9) 输入是否仅改变了形式参数。

(10) 全程变量在各模块中的定义是否一致。

(11) 常数是否当作变量传送。

2. 局部数据结构

局部数据结构与变量的命名、定义、类型、使用等有关,主要检查如下内容。

(1) 不正确或不一致的说明。

(2) 错误的初始化或错误的默认值。

(3) 拼写错或截短的变量名。

(4) 不一致的数据类型。

(5) 上溢、下溢和地址错误。

3. 重要的执行路径

独立路径是至少包括一条新的处理语句或一个新的条件的程序路径,对每一条独立的执行路径至少执行一次。常采用基本路径测试和循环测试,目的是为了发现下面的错误。

(1) 算术运算优先次序不正确或理解错误。例如,对不同的数据类型作比较。

(2) 运算方式不正确。包括逻辑运算不正确或优先次序错误。

(3) 初始化不正确。包括循环不终止或循环终止不正确。

(4) 精度不够。例如,因为精度误差造成本应相等的量不相等。

(5) 表达式的符号表示错误,等等。

4. 错误处理

对于系统出现的错误,主要检查如下内容。

(1) 错误描述难以理解。

(2) 错误提示与实际错误不相符。

(3) 在程序自定义的出错处理段运行之前,系统已介入。

(4) 对错误的处理不正确。

(5) 提供的错误信息不足,无法确定错误位置和查错,等等。

5. 边界问题

边界测试是单元测试步骤中的最后一步,也是最重要的一项任务。众所周知,软件通常容易在边界上失效,因而,采用边界值分析技术,针对边界值及其左、右值设计测试用例,很有可能发现新的错误。主要检查如下内容。

(1) n 重循环的第 0 次、第 1 次和第 n 次是否有错。

(2) n 维数组的第 1 个和第 n 个元素是否有错。

(3) 在运算或判断中的最大取值与最小取值是否有错。

(4) 数据流、控制流或判断条件中刚好小于、等于、大于比较值时是否有错。

通过单元测试可以更早地发现缺陷,缩短开发周期,降低软件开发成本。

5.1.2　单元测试的环境

在进行单元测试时,单元本身无法构成一个完整且切实可行的程序系统。为了执行单元测试,必须为单元测试设计相关的驱动模块和桩模块,才能完成单元测试任务。

1. 驱动模块与桩模块的概念

驱动模块(driver)模拟了被测试模块的上一级模块,相当于被测试模块的主程序。它主要用来接收测试数据,将相关数据传送给被测试模块,并调用被测试模块,打印执行结果。设计驱动模块的目的就是为了访问类库的属性和方法,检测类库的功能是否正确。

桩模块(stub)模拟了被测试模块所调用的模块,不是软件产品的组成部分。在集成测试前要为被测试模块编制一些模拟其下级模块功能的“替身”模块,以代替被测试模块的接口,接收或传递被测试模块的数据,这些专供测试用的“假”模块称为被测试模块的桩模块。

如果被测试的单元模块需要调用其他模块中的功能或者函数,就应设计一个和被调用模块名称相同的桩模块来模拟被调用模块。这个桩模块本身不执行任何功能,仅在被调用时返回静态值来模拟被调用模块的行为。

单元模块的测试环境如图 5-1 所示。

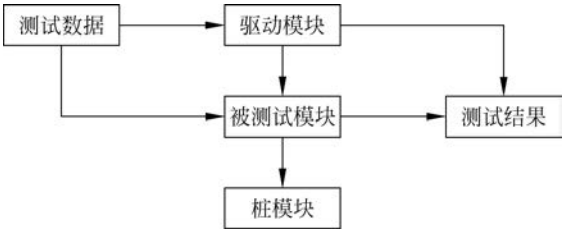

图 5-1　单元模块测试环境

2. 驱动模块与桩模块的设计

由图 5-1 可知,驱动模块接收测试数据并调用被测试模块,最后输出测试结果。所以在

设计驱动模块时,要使驱动模块满足以下这些条件。

(1) 必须能驱动被测试模块的执行。

(2) 能够接收要传递给被测试模块的各项参数,判断其正确性;并将正确的接收数据传送给被测试模块。

(3) 能接收到被测试模块的执行结果,并对结果的正确性进行判断。

(4) 能将判断结果作为测试用例结果并输出测试报告。

桩模块模拟了被测试模块调用的模块,在设计桩模块时,要满足以下这些条件。

(1) 被测试模块必须调用桩模块。

(2) 桩模块必须能正确地接收由被测试模块传递的各项参数,对参数进行正确性判断,并返回执行结果。

(3) 桩模块对外接口的定义必须与被测试模块调用模块的接口一致。

具体要设计多少驱动模块和桩模块呢?这要由具体的系统和采用测试的方法来决定。

例 5-1 假设要对某个系统的部分功能(包括 4 个模块 A、B、C、D)进行测试,其功能分解如图 5-2 所示。

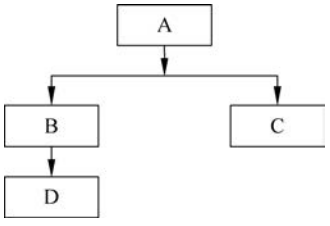

图 5-2 某系统部分功能分解图

若采用非渐进测试方法,那么每一个测试模块都必须设计一个驱动模块。桩模块的设计则由具体的系统来决定。例如,在这个实例中,被测试模块 A 需要设计两个桩模块分别代替 B 和 C 模块,而被测试模块 B 只要设计一个桩模块来代替 D 模块。被测试模块 C、D 不需设计桩模块。

若采用渐增式测试方法,不同渐增方式设计的驱动模块和桩模块也不同。对于采用"自顶向下"的模块测试方法,即测试模块的顺序为 A、B、C、D,那么只要设计一个驱动模块(即测试 A 模块的驱动模块),其他被测试模块用它的上一层模块(已测)作为驱动模块;而需设计的桩模块就多了,与非渐进测试方法一样。若采用"自底向上"的方法,测试模块的顺序是 D、C、B、A。每一个被测试模块都需要设计一个驱动模块,而不需要设计桩模块,各个被测试模块的桩模块用已测试的下层模块直接代替即可。在测试时也可以采用两种方式相结合的方法。

5.1.3 单元测试的过程

单元测试一般由编程人员完成,测试人员可以辅助开发人员进行单元测试。具体过程分为测试计划阶段、测试设计阶段、测试执行阶段和生成测试报告。

1. 测试计划阶段

根据被测试软件的详细设计说明书、代码及测试任务书,对被测试单元进行分析,并确定如下内容。

(1) 确定被测试单元的目标、范围和约束条件。

(2) 确定被测试软件采用的覆盖程度及覆盖的方法和技术。

(3) 确定被测试单元的环境,包括软件、硬件、网络、人员配备等。

(4) 确定被测试单元的测试结束要求。

（5）确定单元测试活动的进度。

2. 测试设计阶段

根据测试计划与要求，对被测试单元设计测试用例，一般由测试人员和测试程序员共同完成。主要工作内容如下。

（1）设计测试用例。

（2）获取测试用例的数据。

（3）确定测试的顺序。

（4）获取测试资源，建立测试环境。

（5）编写测试程序及测试说明文档。

3. 测试执行阶段

根据设计阶段设计好的测试用例，由测试人员对指定的单元进行测试，记录测试步骤及测试结果。主要工作内容如下。

（1）配置单元测试环境。

（2）执行设计阶段的测试用例，并记录执行过程。

（3）记录测试执行结果。

4. 生成测试报告

根据执行阶段产生的测试结果，由测试分析人员进行分析、总结，得到测试结论并写出测试报告。主要完成以下两个方面的工作。

（1）将测试设计中的期望值与实际测试执行结果比较，判定该测试能否通过，并记录结果。

（2）若测试不能通过，分析其不能通过的原因，填写软件问题报告，并提出相关建议。

5.2　单元测试的策略与方法

5.2.1　静态测试与动态测试相结合

单元测试是一种静态与动态相结合的测试。在执行动态测试之前，针对经过编译后的单元测试内容先进行静态代码复审，找出其中的错误。该环节可以由程序设计人员、程序编写人员和程序测试人员参与，由软件设计能力较强的高级程序员任组长，在研究软件设计文档的基础上召开审查会议，分析程序逻辑与错误清单，经测试预演、人工测试、代码复审后再进行计算机代码执行活动的动态测试。所以说，单元测试是静态与动态相结合的测试。

5.2.2　白盒测试与黑盒测试相结合

单元测试主要采用白盒测试方法，辅以黑盒测试方法。其中，白盒测试应用于代码评审、单元程序执行。在白盒测试方法中，以路径覆盖为最佳准则，且系统内多个模块可以并行进行测试。黑盒测试方法则应用于模块、组件等大单元的功能测试。

5.2.3 人工测试与自动化测试相结合

人工测试是由测试人员手工逐步执行所有的活动,并观察每一步是否成功完成。人工测试是测试活动的必要部分,在软件及其用户接口还未足够稳定的开发初始阶段尤其有效。

在不能使用自动化测试工具时,必须采用人工测试的方法对单元相关内容进行测试。

自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程。在设计了测试用例并通过评审之后,由测试人员采用自动化测试工具,根据测试用例中描述的规程一步步执行测试,得到实际结果与期望结果的比较。在此过程中,可以节省人力、时间或硬件资源,提高测试效率。

单元自动化测试工具有 JUnit、C++Test、JFCUnit、VSTS 等。

5.3 集成测试的概述

5.3.1 集成测试的定义

集成测试也叫组装测试或联合测试。在单元测试的基础上,将所有模块按照设计要求组装成为子系统或系统,进行集成测试。

集成测试是单元测试的逻辑扩展。在现实方案中,集成是指多个单元的聚合,许多单元组合成模块,而这些模块又聚合成程序的更大部分,如分系统或系统。集成测试采用的方法是测试软件单元的组合能否正常工作,以及与其他组的模块能否集成起来工作。最后,还要测试构成系统的所有模块组合能否正常工作。集成测试所持的主要标准是《软件概要设计规格说明》,任何不符合该说明的程序模块行为都应该加以记载并上报。

集成测试关注的主要内容如下。

(1) 模块接口的数据交换;

(2) 各子功能组合起来能否达到预期要求的父功能;

(3) 模块间是否有不利影响;

(4) 全局数据结构是否有问题;

(5) 单个模块的误差是否会累积放大。

5.3.2 集成测试的目标

集成测试的目标是按照设计要求使用那些通过单元测试的构件来构造程序结构。单个模块具有高质量并不足以保证整个系统的高质量。有许多隐蔽的失效是高质量模块间发生非预期交互而产生的。

判断集成测试是否完成,可从以下几个方面进行检查。

(1) 成功地执行了测试计划中规定的所有集成测试;

(2) 修正了所发现的错误;

(3) 测试结果通过了专门小组的评审。

集成测试应由专门的测试小组进行,测试小组由有经验的系统设计人员和程序员组成。整个测试活动要在评审人员出席的情况下进行。

在完成预定的组装测试工作之后，测试小组应负责对测试结果进行整理、分析，形成测试报告。测试报告中要记录实际的测试结果、在测试中发现的问题、解决这些问题的方法以及解决之后再次测试的结果。此外还应写明不能解决、还需要管理人员和开发人员注意的一些问题，提供测试评审和最终决策，以便提出处理意见。

5.4　集成测试的方法

5.4.1　大爆炸集成测试

1. 大爆炸集成测试概述

大爆炸集成也称为一次性组装或整体拼装，是一种非增量式组装方式。这种集成测试策略的做法就是把所有通过单元测试的模块一次性集成到一起进行测试，不考虑组件之间的互相依赖性及可能存在的风险。

例 5-2　某个系统组成模块如图 5-3 所示，包括模块 A、B、C、D、E、F、G。

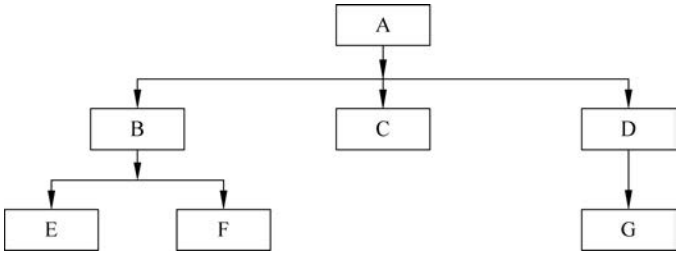

图 5-3　某个系统的层次模块图

采用大爆炸集成测试方法，首先对各个模块（A、B、C、D、E、F、G）分别进行单元测试，然后再把所有模块组装在一起进行测试。

2. 大爆炸集成测试的优点

大爆炸集成测试具有如下优点。

（1）可以并行测试所有模块。

（2）需要的测试用例数目少。

（3）测试方法简单、易行。

3. 大爆炸集成测试的缺点

大爆炸集成测试具有如下缺点。

（1）因为不可避免地存在模块间接口、全局数据结构等方面的问题，所以一次运行成功的可能性不大。

（2）如果一次集成的模块数量多，集成测试后可能会出现大量的错误。另外，修改了一处错误之后，很可能新增更多的新错误，新旧错误混杂，给程序的错误定位与修改带来很大的麻烦。

（3）即使集成测试通过，也会遗漏很多错误。

4. 大爆炸集成测试的适用范围

大爆炸集成测试适用于如下范围。

(1) 只需要修改或增加少数几个模块的前期产品稳定的项目。

(2) 功能少,模块数量不多,程序逻辑简单,并且每个组件都已经过充分单元测试的小型项目。

(3) 基于严格的净室软件工程(由IBM公司开创的开发接近零缺陷的软件的成功做法)开发,并且在每个开发阶段,产品质量和单元测试质量都相当高的产品。

5.4.2 自顶向下集成测试

1. 自顶向下集成测试概述

自顶向下的集成测试就是按照系统层次结构图,以主程序模块为中心,从顶层控制(主控模块)开始,自上而下按照深度优先或者广度优先策略,对各个模块一边组装一边进行测试。采用同设计顺序一样的思路对被测系统进行测试,来验证系统的功能性和稳定性。

2. 自顶向下集成测试的步骤

自顶向下集成测试的测试步骤如下。

(1) 先对主控模块进行测试,用桩模块代替所有直接附属于主控模块的模块。

(2) 根据选定的优先策略(广度或深度优先),每次用一个实际模块代替一个桩模块进行测试。

(3) 在结合下一模块的同时进行测试。

(4) 为了保证加入的模块没有引入新的错误,需要进行回归测试。

(5) 重复步骤(2)~(4),直到所有的模块集成测试完成。

例 5-3 对于图 5-3 中的模块,将其分为三个层次,上层包括模块 A、中层包括模块 B、C、D,下层包括模块 E、F、G。

采用广度优先自顶向下测试方法:首先测试模块 A,其次将模块 A、B、C、D 集成测试,最后将所有模块 A、B、C、D、E、F、G 集成测试。

采用深度优先自顶向下测试方法:首先测试模块 A,其次将模块 A、B 集成,A、B、E、F 集成测试;再将 A、B、E、F、C 集成;然后将 A、B、E、F、C、D 集成,最后对 A、B、E、F、C、D、G 进行集成测试。

3. 自顶向下集成测试的优点

自顶向下集成测试的主要优点如下。

(1) 较早验证了主要的控制和判断点。

(2) 功能可行性较早得到证实。

(3) 最多只需要一个驱动模块。

(4) 可以与设计并行进行测试。

(5) 支持故障隔离。

4. 自顶向下集成测试的缺点

自顶向下集成测试的主要缺点如下。

(1) 桩开发和维护的成本大。

(2) 底层组件的一个需求的修改会导致许多顶层组件的修改。

(3) 底层模块越多,越会导致底层测试不充分。

5. 自顶向下集成测试的适用范围

自顶向下集成测试主要适合于采用结构化编程方法,且产品结构相对简单的软件产品。

5.4.3　自底向上集成测试

1. 自底向上集成测试概述

自底向上集成是从系统层次结构图的最底层模块开始,按照层次结构图逐层向上进行组装和集成测试的方式。

2. 自底向上集成测试的步骤

自底向上集成测试的测试步骤如下。

(1) 从最底层的模块开始组装测试。

(2) 编写驱动程序,协调测试用例的输入与输出。

(3) 测试集成后的构件。

(4) 使用实际模块代替驱动程序,按程序结构向上组装测试后的构件。

(5) 重复步骤(2)～(4),直到系统的最顶层模块被加入到系统中完成测试为止。

例 5-4　对于图 5-3 中的模块,采用自底向上方法进行测试。

对分别已进行了单元测试的各个模块,先分别对集成 B、E、F 和集成 D、G 并行进行测试。需要编写各自的驱动模块,最后将模块 A、B、C、D、E、F、G 集成测试。

3. 自底向上集成测试的优点

自底向上集成测试的优点如下。

(1) 较早验证底层模块。

(2) 工作最初可以并行集成测试,集成策略小。

(3) 减少桩模块编写的工作量,支持故障隔离。

4. 自底向上集成测试的缺点

自底向上集成测试的缺点如下。

(1) 驱动模块开发工作量大。

(2) 对高层的验证在最后,设计上的错误不能被及时发现。

5. 自底向上集成测试的适用范围

自底向上集成测试适用于大部分采用结构化编程方法,且产品结构相对简单的软件产品。

5.4.4　三明治集成测试

1. 三明治集成测试概述

三明治集成是一种混合增殖式测试策略,综合了自顶向下和自底向上两种集成方法。

它把系统划分成三层,中间一层为目标层,目标层以上采用自顶向下集成,目标层以下采用自底向上集成。

例 5-5 对于图 5-3 中的模块,采用三明治集成方法进行测试。

对分别已进行了单元测试的各个模块,在下层分别集成模块 B、E、F 和 D、G 进行测试,在上层集成模块 A、B、C、D 进行测试,最后将所有模块集成测试。

2. 三明治集成测试的优点

三明治集成测试的优点是:集合了自顶向下和自底向上两种集成测试的优点。

3. 三明治集成测试的缺点

三明治集成测试的缺点是:由于中间层是最后集成测试,因此中间层在被集成前测试不充分。

4. 三明治集成测试的适用范围

三明治集成测试对大部分开发软件项目都适用。

5.4.5 其他集成测试策略

1. 核心系统先行集成测试

核心系统先行集成测试法的思想是先对核心软件部件进行集成测试,在测试通过的基础上再按各外围软件部件的重要程度逐个集成到核心系统中。每次加入一条外围软件部件都产生一个产品基线,直至最后形成稳定的软件产品。核心系统先行集成测试法对应的集成过程是一条逐渐趋于闭合的螺旋形曲线,代表产品逐步定型的过程。

该集成测试方法对于快速软件开发很有效果,适合较复杂系统的集成测试,能保证一些重要的功能和服务的实现。缺点是限制较多:采用此法的系统一般应能明确区分核心软件部件和外围软件部件,核心软件部件应具有较高的耦合度,外围软件部件内部也应具有较高的耦合度,但各外围软件部件之间应具有较低的耦合度。

2. 高频集成测试

高频集成测试是指同步于软件开发过程,每隔一段时间对开发团队的现有代码进行一次集成测试。例如,某些自动化集成测试工具能实现每日深夜对开发团队的现有代码进行一次集成测试,然后将测试结果发到各开发人员的电子邮箱中。该集成测试方法频繁地将新代码加入到一条已经稳定的基线中,以免集成故障难以发现,同时控制可能出现的基线偏差。

使用高频集成测试需要具备一定的条件:可以持续获得一个稳定的增量,并且该增量内部已被验证没有问题;大部分有意义的功能增加可以在一个相对稳定的时间间隔(如每个工作日)内获得;测试包和代码的开发工作必须是并行进行的,并且需要版本控制工具来保证始终维护的是测试脚本和代码的最新版本;必须借助于自动化工具来完成。高频集成的一个显著特点就是集成次数频繁,显然,人工测试的方法是不能胜任的,必须采用自动化测试工具。

该测试方案能在开发过程中及时发现代码错误,能直观地看到开发团队的有效工程进

度。在此方案中，开发维护源代码与开发维护软件测试包被赋予了同等的重要性，这对有效防止错误、及时纠正错误都有帮助。该方案的缺点在于测试包有时候可能不能暴露深层次的编码错误和图形界面错误。

3. 基于功能的集成测试

基于功能的集成测试从功能角度出发，按照功能的关键程度对模块的集成顺序进行组织，尽可能早地进行测试。该方法关注测试验证系统的关键功能。

此方法适用于对有较大风险的产品、技术探索性项目或者是对功能实现没有把握的产品进行测试，被测试产品功能的实现比质量更关键。

4. 基于进度的集成测试

基于进度的集成测试主要是从系统的进度和质量两方面考虑，尽早地进行集成测试，以提高开发和集成的并行性，有效地缩短项目的开发时间，提高开发项目的质量。但此方法早期测试缺乏整体性，仅能进行独立的集成，导致许多接口要到后期才能验证。桩模块与驱动模块的开发量大，模块可能不稳定、产生变化，导致测试的重复与浪费。

5. 基于风险的集成测试

基于风险的集成测试是基于一种假设：系统的错误往往集中在系统风险最高的模块中。因此应对高风险的模块接口先进行重点测试，从而保证系统的稳定性。

尽早验证高风险的接口有助于加速系统的稳定性，有利于加强对系统的信心。此方法可以与功能集成测试结合起来使用，主要适用于系统中风险较大的模块测试。

6. 客户/服务器的集成测试

客户/服务器的集成测试主要针对客户/服务器系统，对系统客户端与服务器端交互进行集成测试。先单独测试每个客户端和服务器端，再将第一个客户端与服务端集成测试，加入下一个客户端与服务端集成测试。如此循环，将所有客户端与服务器端集成测试完成。

此方法对集成次序没有约束，有利于复用和扩充；支持可控制和可重复的测试。但驱动模块与桩模块的开发成本高。

5.5　集成测试阶段的测试过程

根据 IEEE 标准，集成测试过程划分为 5 个阶段：计划阶段、设计阶段、实施阶段、执行阶段以及评估阶段。

5.5.1　集成测试计划阶段

1. 集成测试准备

在进行集成测试前，先要准备如下内容。

(1) 测试的文档准备：制定需求规格说明书、概要设计文档、产品开发计划。

(2) 人员组织：任命测试人员、开发人员、测试质量控制人员、测试经理、开发经理、产品经理。

各相关人员职责如表5-1所示。

表5-1 集成测试中相关人员及职责

角　色	职　责
测试人员	负责测试用例设计、执行、记录、回归测试
开发人员	负责定位和解决问题,修复缺陷
测试质量控制人员	负责对集成测试进行监控、评审
测试经理	负责制定测试计划、安排测试任务
开发经理	负责代码的修复和安排管理
产品经理	负责解决资源要求(人力资源、工具等),并对测试结果进行监督

2. 集成测试策略和环境

集成测试策略考虑采用的测试技术和工具,完成测试影响的资源分配及其他特殊的问题。

在环境方面的考虑如下。

(1) 硬件环境:尽可能考虑实际环境。

(2) 操作系统环境:不同机型使用不同的操作系统版本。

(3) 数据库环境:从性能、版本、容量等多方面考虑。

(4) 网络环境。

3. 测试日程计划

根据软件设计文档评估测试有多少项目,再根据测试的工作量进行安排。开始时间为概要设计完成评审后大约一个星期。

4. 活动步骤

集成测试的主要活动步骤如下。

(1) 确定被测试对象和测试范围。

(2) 评估集成测试被测试对象的数量及难度,即工作量。

(3) 确定角色分工和任务。

(4) 标识出测试各阶段的时间、任务、约束等条件。

(5) 考虑一定的风险分析及应急计划。

(6) 考虑和准备集成测试需要的测试工具、测试仪器、环境等资源。

(7) 考虑外部技术支援的力度和深度,以及相关培训安排。

(8) 定义测试完成的标准。

5. 输出

集成测试计划阶段最后得到集成测试计划,此计划必须通过概要设计阶段基线评审。

5.5.2 集成测试设计阶段

1. 时间安排

集成测试的设计在系统详细设计阶段就可以开始。

2. 依据

集成测试设计阶段的主要依据是：需求规格说明书、概要设计、集成测试计划。

3. 入口条件

系统的概要设计基线已通过评审。

4. 活动步骤

集成测试设计阶段具体的活动步骤如下。

(1) 被测对象结构分析。

(2) 集成测试模块分析。

(3) 集成测试接口分析。

(4) 集成测试策略分析。

(5) 集成测试工具分析。

(6) 集成测试环境分析。

5. 输出

集成测试设计阶段的输出是集成测试设计方案。

6. 出口条件

集成测试设计通过详细设计基线评审。

5.5.3　集成测试实施阶段

根据集成测试计划，建立集成测试环境，完成测试设计任务。

1. 时间安排

在系统编码阶段开始后就可以实施集成测试。

2. 依据

系统的需求规格说明书、概要设计、集成测试计划、集成测试设计计划。

3. 入口条件

系统的详细设计基线通过评审。

4. 活动步骤

集成测试实施阶段的活动步骤如下。

(1) 集成测试用例设计。

(2) 集成测试代码设计(如果需要)。

(3) 集成测试脚本设计(如果需要)。

(4) 集成测试工具准备(如果需要)。

5. 输出

集成测试设计的最后输出有集成测试用例、集成测试规程、集成测试代码、集成测试脚本、集成测试工具。

测试用例模板如表 5-2 所示。

表 5-2 集成测试用例模板

测试任务编号		测试任务描述		
作　　者		日　　期		
设计方法		测试用例脚本文件名		
预置条件				
用例编号	输入	执行动作	预期结果	备注

6. 出口条件

集成测试设计阶段的出口条件是：测试用例和测试规程通过编码阶段基线评审。

5.5.4 集成测试执行阶段

按照集成测试用例设计要求构建平台。

1. 时间安排

系统的单元测试完成后就可以开始执行系统的集成测试了。

2. 输入

集成测试需要的基本内容：需求规格说明书、概要设计、集成测试计划、集成测试用例、集成测试规程、集成测试代码(如果有)、集成测试脚本、集成测试工具、详细设计代码、单元测试报告。

3. 入口条件

系统的单元测试阶段已经通过基线化评审。

4. 活动步骤

集成测试实施阶段的活动步骤如下。

(1) 执行集成测试用例。

(2) 回归集成测试用例。

(3) 撰写集成测试报告。

5. 输出

集成测试实施后，生成集成测试报告。

6. 出口条件

集成测试报告通过集成测试阶段基线评审。

5.5.5 集成测试评估阶段

由测试设计员负责，与集成测试人员、编码员、设计员等对集成测试结果进行统计，生成测试执行报告和缺陷记录报告；对集成测试进行评估，对测试结果进行评测，形成结论，最后整理形成报告。

5.6　集成测试与单元测试的比较

5.6.1　测试的单元不同

单元测试是针对软件的基本单元(如函数等)所做的测试,主要测试单元的功能和性能等;集成测试则是以模块和子系统为单元进行的测试,主要测试模块接口间的关系。

5.6.2　测试的依据不同

单元测试是针对软件的详细设计做的测试,测试用例的主要依据也是系统的详细设计。集成测试是针对软件的概要设计做的测试,测试用例的主要依据则是系统的概要设计。

5.6.3　测试的空间不同

集成测试主要测试的是接口层的测试空间,单元测试主要测试的是内部实现层的测试空间。

5.6.4　测试使用的方法不同

集成测试关注的是接口的集成,而单元测试只关注单个单元,因此在具体测试方法上也不同。

5.7　系统测试概述

5.7.1　系统测试定义和技术要求

1. 系统测试定义

系统测试是将集成好的软件系统整体作为基于计算机系统的一个元素,与计算机硬件、外设、支持软件、数据等其他系统元素结合在一起,在实际运行(使用)环境下所进行的一系列测试活动。

2. 系统测试的目的

通过与系统的需求定义比较,检查软件是否存在与系统定义不符合或与之矛盾的地方,以验证软件系统的功能和性能等是否满足其规约所指定的要求。

3. 系统测试技术要求

系统测试的基本技术要求如下。

(1) 系统的每个特性应至少被一个正常测试用例和一个被认可的异常测试用例所覆盖。

(2) 测试用例的输入应至少包括有效等价类值、无效等价类值和边界数据值。

(3) 应逐项测试系统/子系统设计说明规定的系统的功能、性能等特性。

(4) 应测试软件配置项之间及软件配置项与硬件之间的接口。

(5) 应测试系统的输出及其格式。

(6) 应测试当运行条件在边界状态和异常状态下时,或在人为设定的状态下时,系统的功能和性能。

(7) 应测试系统访问和数据安全性。

(8) 应测试系统的全部存储量、输入/输出通道和处理时间的余量。

(9) 应按系统或子系统设计文档的要求,对系统的功能、性能进行强度测试。

(10) 应测试设计中用于提高系统安全性、可靠性的结构、算法、容错、冗余、中断处理等方案。

(11) 对完整性级别高的系统应进行安全性、可靠性分析,明确每一个危险状态和导致危险的可能原因,并对其进行针对性的测试。

(12) 对有恢复或重置功能需求的系统,应测试其恢复或重置功能和平均恢复时间,并且对每一类导致恢复或重置的情况进行测试。

(13) 对不同的实际问题应外加相应的专门测试。

5.7.2 系统测试的内容

国标 GB/T 16620 针对系统测试的测试内容主要从适应性、准确性、互操作性、安全保密性、成熟性、容错性、易恢复性、易理解性、易学性、易操作性、吸引性、时间特性、资源利用性、易分析性、易改变性、稳定性、易测试性、适应性、易安装性、共存性、替换性和依从性等方面(有选择地)来考虑。

对于具体的系统,可根据测试合同(或项目计划)及系统/子系统设计文档的要求对上述测试内容进行剪裁。

1. 功能性

系统功能性方面的主要测试内容如下。

(1) 适应性方面:应测试系统/子系统设计文档规定的系统的每一项功能。

(2) 准确性方面:可对系统中具有准确性要求的功能和精度要求的项(如数据处理精度、时间控制精度、时间测量精度)进行测试。

(3) 互操作性方面:可测试系统/子系统设计文档、接口需求规格说明文档和接口设计文档规定的系统与外部设备的接口、与其他系统的接口。测试其格式和内容,包括数据交换的数据格式和内容;测试接口之间的协调性;测试软件系统每一个真实接口的正确性;测试软件系统从接口接收和发送数据的能力;测试数据的约定、协议的一致性;测试软件系统对外围设备接口特性的适应性。

(4) 安全保密性方面:可测试系统及其数据访问的可控制性;测试系统防止非法操作的模式,包括防止非授权的创建、删除或修改程序或信息的行为,必要时可做强化异常操作的测试;测试系统防止数据被讹误和被破坏的能力;测试系统的加密和解密功能。

2. 可靠性

系统可靠性方面的主要测试内容如下。

(1) 成熟性方面:可基于系统运行剖面设计测试用例,根据实际使用的概率分布随机选

择输入，运行系统，测试系统满足需求的程度并获取失效数据，其中包括对重要输入变量值的覆盖、对相关输入变量可能组合的覆盖、对设计输入空间与实际输入空间之间区域的覆盖、对各种使用功能的覆盖、对使用环境的覆盖。应在有代表性的使用环境中以及可能影响系统运行方式的环境中运行软件，验证系统的可靠性需求是否正确实现。对一些特殊的系统，如容错软件、实时嵌入式软件等，由于在一般的使用环境下常常很难在软件中植入差错，应考虑多种测试环境。可测试系统的平均无故障时间。选择可靠性增长模型，通过检测到的失效数和故障数，对系统的可靠性进行预测。

（2）容错性方面：可测试系统对中断发生的反应；系统在边界条件下的反应；系统的功能、性能的降级情况；系统的各种误操作模式；系统的各种故障模式（如数据超范围、死锁）；测试在多机系统出现故障需要切换时，系统的功能和性能的连续平稳性。

注：可用故障树分析技术检测误操作模式和故障模式。

（3）易恢复性方面：可测试具有自动修复功能的系统的自动修复时间；系统在特定时间范围内的平均宕机时间；系统在特定时间范围内的平均恢复时间；系统的重新启动并继续提供服务的能力；系统的还原能力。

3. 易用性

系统易用性方面的主要测试内容如下。

（1）易理解方面：对于系统的各项功能，确认它们是否容易被识别和理解。若系统要求具有演示功能的能力，应确认演示是否容易被访问、演示是否充分和有效。对于界面的输入和输出，应确认输入和输出的格式和含义是否容易被理解。

（2）易学性方面：可测试系统的在线帮助功能，确认在线帮助是否容易定位，是否有效；还可以对照用户手册或操作手册执行系统，测试用户文档的有效性。

（3）易操作性方面：输入数据，确认系统是否对输入数据进行有效性检查。对于要求具有中断执行的功能，应确认它们能否在动作完成之前被取消。对于要求具有还原能力（数据库的事务回滚能力）的功能，应确认它们能否在动作完成之后被撤销。对于包含参数设置的功能，应确认参数是否已选择、是否有默认值。对于要求具有解释的消息，应确认它们是否明确。对于要求具有界面提示能力的界面元素，应确认它们是否有效。对于要求具有容错能力的功能和操作，应确认系统能否提示出错的风险，能否容易纠正错误的输入，能否从差错中恢复。对于要求具有定制能力的功能和操作，应确认定制能力的有效性。对于要求具有运行状态监控能力的功能，应确认它们的有效性。

注：以正确操作、误操作模式、非常规模式和快速操作为框架设计测试用例，误操作模式有错误的数据类型作参数、错误的输入数据序列、错误的操作序列等。如有用户手册或操作手册，可对照手册逐条进行测试。

（4）吸引性方面：可测试系统的人机交互界面能否定制。

4. 效率

系统效率方面的主要测试内容如下。

（1）时间特性方面：可测试系统的响应时间、平均响应时间、响应极限时间，系统的吞吐量、平均吞吐量，系统的周转时间、平均周转时间、周转时间极限。

注：响应时间指系统为完成一项规定任务所需的时间；平均响应时间指系统执行若干并行任务所需的平均时间；响应极限时间指在最大负载条件下，系统完成某项任务需要时间的极限；吞吐量指系统在给定的时间周期内能成功完成的任务数量；平均吞吐量指系统在一个单位时间内能处理并发任务的平均数；极限吞吐量指在最大负载条件下，在给定的时间周期内，系统能处理的最多并发任务数；周转时间指从发出一条指令到完成一组相关的任务的时间；平均周转时间指完成一个任务所需要的平均时间；周转时间极限指在最大负载条件下，系统完成一线任务所需要时间的极限。

在测试时，应标识和定义适合于软件应用的任务，并对多项任务进行测试，而不是仅测一项任务。

注：软件应用任务的例子包括在通信应用中的切换、数据包发送、在控制应用中的事件控制、在公共用户应用中由用户调用的功能产生的一个数据的输出等。

(2) 资源利用性方面：可测试系统的输入/输出设备、内存和传输资源的利用情况，执行大量的并发任务，测试输入/输出设备的利用时间；在使输入/输出负载达到最大的系统条件下，运行系统，测试输入/输出负载极限；并发执行大量的任务，测试用户等待输入/输出设备操作完成需要的时间。

注：建议调查几次测试与运行实例中的最大时间与时间分布。在规定的负载下和在规定的时间范围内运行系统，测试内存的利用情况；在最大负载下运行系统，测试内存的利用情况；并发执行规定的数个任务，测试系统的传输能力；在系统负载最大的条件下和在规定的时间周期内，测试传输资源的利用情况；在系统传输负载最大的条件下，测试不同介质同步完成其任务的时间周期。

5. 维护性

系统维护性方面的主要测试内容如下。

(1) 易分析性方面：可设计各种情况的测试用例，运行系统并监测系统运行状态数据，检查这些数据是否容易获得，内容是否充分。若软件具有诊断功能，则应测试该功能。

(2) 易改变性方面：可测试能否通过参数来改变系统。

(3) 易测试性方面：可测试软件内置的测试功能，确认它们是否完整和有效。

6. 可移植性

系统可移植性方面的主要测试内容如下。

(1) 适应性方面：可测试软件对数据文件、数据块或数据库等数据结构的适应能力；软件对硬件设备和网络设施等硬件环境的适应能力；软件对系统软件或并行的应用软件等软件环境的适应能力；软件是否已移植。

(2) 易安装性方面：可测试软件安装的工作量、安装的可定制性、安装设计的完备性、安装操作的简易性、是否容易重新安装。

注：安装设计的完备性可分为以下三级。

① 最好：设计了安装程序并编写了安装指南文档。

② 好：仅编写了安装指南文档。

③ 差：无安装程序和安装指南文档。

注：安装操作的简易性可分为以下四级。

① 非常容易：只需启动安装功能并观察安装过程。

② 容易：只需回答安装功能中提出的问题。

③ 不容易：需要从表或填充框中看参数。

④ 复杂：需要从文件中寻找参数，改变或写入参数。

(3) 共存性方面：可测试软件与其他软件共同运行的情况。

(4) 易替换性方面：当替换整个不同的软件系统和用同一软件系列的高版本替换低版本时，在易替换性方面，可考虑测试：

① 软件能否继续使用被其替代的软件使用过的数据；

② 软件是否具有被其替代的软件中的类似功能。

(5) 依从性方面：当软件在功能性、可靠性、易用性、效率、维护性和可移植性方面遵循了相关的标准、约定、风格指南或法规时，应酌情进行测试。

5.8　系统测试的方法与过程

5.8.1　系统测试方法

1. 功能测试

对产品的功能进行测试，检验是否实现、是否正确实现系统功能。

2. 性能测试

对产品的性能进行测试，检验是否达标、是否能够保持性能。

3. 负载测试

在人为设置的高负载(大数据量、大访问量)的情况下，检查系统是否出现功能或者性能上的问题。

4. 压力测试

在人为设置的系统资源紧缺情况下，检查系统是否出现功能或者性能上的问题。

5. 疲劳测试

在一段时间内(经验上一般是连续 72 小时)保持系统的频繁使用，检查系统是否出现功能或者性能上的问题。

6. 易用性测试

检查系统界面和功能是否容易学习，使用方式是否规范一致，是否会误导用户或者使用了模糊的信息。

7. 安装测试

检查系统安装是否能够安装所有需要的文件/数据并进行必要的系统设置，检查系统安装是否会破坏其他文件或配置，检查系统安装是否可以中止并恢复现场，检查系统是否能够正确卸载并恢复现场，检查安装和卸载过程的用户提示和功能是否出现错误。有时候将安

装测试作为功能测试的一部分。

8. 配置测试

在不同的硬件配置下,在不同的操作系统和应用软件环境中,检查系统是否发生功能或者性能上的问题。

9. 文档测试

检查系统的文档是否齐全,检查是否有多余文档或者死文档,检查文档内容是否正确/规范/一致等。

10. 安全测试

检查系统是否有病毒,检查系统是否正确加密,检查系统在非授权的内部或外部用户访问或故意破坏时是否出现错误。

11. 恢复测试

在人为发生系统灾难(系统崩溃、硬件损坏、病毒入侵等)的情况下,检查系统是否能恢复被破坏的环境和数据。

12. 回归测试

回归测试是一种选择性重新测试,目的是检测系统或系统组成部分在修改期间产生的缺陷,用于验证已进行的修改并未引起不希望的有害效果,或确认修改后的系统或系统组成部分仍满足规定的要求。

13. 健全测试

检查系统的功能和性能是否基本可以正常使用,确定是否可以继续进行系统测试的其他内容。

14. 交付测试

关闭所有缺陷报告,确保系统达到预期的交付标准。

15. 演练测试

在交付给用户之前,利用相似的用户环境进行测试。例如,奥运会管理信息系统在2008年前用于其他比赛。

16. 背靠背测试

设置一组以上的测试团队,在互相不进行沟通的情况下独立进行相同的测试项目,用来评估测试团队的效果并发现更多的错误。该方法开始用于测试外包,现在也用于内部测试。

17. 度量测试

在系统中人为地放入错误(播种),并根据被发现的比例来确定系统中遗留的错误数量。该方法开始用于测试外包,现在也用于内部测试。

18. 比较测试

与竞争产品及本产品的旧版本测试同样的内容,以确定系统的优势和劣势。严格地说,比较测试属于系统测评的内容。BenchMarking是一种特殊的比较测试。

上述 18 种测试内容并不是都要进行的。在制定测试策略和测试计划的时候，要有不同的侧重点，且与测试目标、测试资源、软件系统特点和业务环境有关。

另外，上述 18 种测试最好由独立第三方进行测试，因为进行独立测试的目的是进一步加强软件质量保证工作，提高软件的质量，并对软件产品进行客观评价。由独立第三方测试通常能够发挥专业技术优势和独立性优势，能够有效地促进承办方的工作。

5.8.2　系统测试过程

系统测试过程主要包含 4 个阶段：制定系统测试计划、设计系统测试用例、执行系统测试、提交系统测试报告。

1. 制定系统测试计划

系统测试计划是软件测试员与产品开发小组交流的主要方式。测试小组共同协商测试计划，测试组长按照测试模板起草《系统测试计划》。其主要内容有：规定测试活动的范围、测试方法、资源（测试环境、测试辅助工具）与进度；明确正在测试的项目、要测试的特性、要执行的测试任务、每个任务的负责人，以及与计划相关的内容。

项目经理审批《系统测试计划》后，进行到下一个阶段。

2. 设计系统测试用例

系统测试小组成员依据《系统测试计划》和指定的模板设计《系统测试用例》。其中包括三个部分：测试设计说明、测试用例说明、测试程序和测试过程说明。

1）测试设计说明

IEEE 829 标准提出，测试设计说明“在测试计划中提炼测试方法，明确指出设计包含的特性及其相关测试，如果要求完成测试还应明确指出测试用例和测试程序，指定判断特性通过/失败的规则。”

测试设计说明的目的是组织和描述针对具体特性需要进行的测试。其主要内容如下。

（1）标识符：用于引用和标记测试设计说明的唯一标识符。

（2）要测试的特性：对测试设计说明所包含的软件特性的描述。

（3）方法：描述测试软件特性的通用方法。

（4）测试用例确认：对所测试特性的具体测试用例的高级描述和引用。它应列出所选的等价划分，并提供测试用例的引用信息以及用于执行测试用例的程序。

（5）通过/失败规则：描述判定测试特性通过或失败的规则。例如，通过是指执行全部测试用例时没有发现软件缺陷；失败是指有 10％以上测试用例没有通过。

2）测试用例说明

IEEE 829 标准提出，测试用例说明需要“编写用于输入的实际数值和预期输出结果数值，明确指出使用具体测试用例产生的测试程序的限制。”

测试用例应该明确地解释要向软件发送的数据或限制的条件，以及预期的结果。测试用例可以由一个或多个测试用例说明引用，也可以引用多个测试程序。其主要内容如下。

（1）标识符：由测试设计过程说明和测试程序说明引用的唯一标识符。

（2）测试项：描述被测试的详细特性、代码模块等，比设计说明中所列的特性更具体。

(3) 输入说明：列举软件执行测试用例的所有输入内容或者条件。

(4) 输出说明：描述执行测试用例的预期结果。

(5) 环境要求：是指执行测试用例的硬件、软件、工具、人员等。

(6) 特殊过程要求：描述执行测试必须做到的特殊要求。

(7) 用例之间的依赖性：说明测试用例是否与其他测试用例有依赖关系。

3) 测试程序和过程说明

IEEE 829 标准定义测试程序是“明确指出为实现相关测试设计而操作软件系统和试验具体测试用例的全部步骤。”

测试程序或测试脚本说明详细定义了执行测试用例的每一步操作。主要内容如下。

(1) 标识符：将测试程序与相关测试用例和测试设计捆绑在一起的唯一标识符。

(2) 目的：程序的目的以及将要执行的测试用例的引用信息。

(3) 特殊要求：执行程序所需的其他程序、特殊测试技术或者特殊设备。

(4) 程序步骤：执行测试的详细描述，包括日志、设置、启动、程序、度量、关闭、重启、终止、重置、偶然事件等。

测试组长邀请开发人员和同行专家对测试用例进行技术评审，通过后进行下一阶段。

3. 执行系统测试

系统测试人员依据《系统测试计划》和《系统测试用例》对系统进行测试，将执行结果记录在《系统测试报告》中，并及时将存在的缺陷通报给开发人员。开发人员及时改正已发现的缺陷，并由测试人员进行回归测试，以确保不会引入新的缺陷。

软件缺陷报告包括的主要内容如下。

(1) 标识符：定义软件缺陷报告的唯一编号，用于定位和引用。

(2) 总结：利用简明扼要的事实陈述总结软件缺陷。要测试的软件及版本的引用信息，相关的测试过程、测试用例和测试说明等。

(3) 事件描述：提供软件缺陷的详细描述信息。主要包括日期和时间、测试员姓名、使用的硬件和软件配置、输入、过程步骤、预期结果、试图再现以及尝试的描述等。

(4) 影响：严重性和优先级，以及测试计划、测试说明、测试程序和测试用例的影响指示。

4. 提交系统测试报告

系统所有的测试执行完成后，提交测试报告文档。

测试报告模板的主要内容如下。

1 引言

1.1 编写的目的

1.2 编写的背景

1.3 术语解释

1.4 参考资料

2 测试概要

2.1 系统简介

2.2 测试计划描述

2.3　测试环境
3　测试结果及分析
3.1　测试执行情况
3.2　测试功能报告
3.2.1　系统管理模块测试报告单
3.2.2　功能插件模块测试报告单
3.2.3　网站管理模块测试报告单
3.2.4　内容管理模块测试报告单
3.2.5　辅助工具模块测试报告单
3.3　系统性能测试报告
3.4　不间断运行测试报告
3.5　易用性测试报告
3.6　安全性测试报告
3.7　可靠性测试报告
3.8　可维护性测试报告
4　测试结论与建议
4.1　测试人员对需求的理解
4.2　测试准备和测试执行过程
4.3　测试结果分析
4.4　建议

小结

本章介绍了单元测试、集成测试和系统测试，包括：单元测试的主要任务和测试环境的搭建，单元测试中驱动模块与桩模块的设计，单元测试的测试过程，单元测试的策略与方法；集成测试的基本概念，集成测试的方法（大爆炸集成、自顶向下集成、自底向上集成、三明治集成、高频集成等），集成测试的测试过程；系统测试的基本概念，系统测试的测试内容、方法与过程，以及测试报告的撰写。

习题

1. 单元测试的主要任务有哪些？请简述。
2. 怎样搭建单元测试的环境？
3. 单元测试的过程分为哪几个阶段？请简述。
4. 集成测试的方法有哪几种？集成测试包括哪些阶段？
5. 系统测试主要测试哪些方面？
6. 单元测试、集成测试与系统测试有什么区别？能否缺少其中某种测试？