* + 1. 信息系统测试

系统测试是保证管理信息系统质量的一个重要环节。程序编制完成后，要用各种测试方法检查各个部分是否达到了规定的质量标准。系统测试是为了发现程序和系统中的错误。好的测试方案有可能发现从未发现的错误，能够发现从未发现过的错误的测试才是成功的测试，否则就没有必要进行测试了。

6.3.2系统测试概述

1. 系统测试概述

系统测试，英文是System Testing。是将已经确认的[软件](http://baike.baidu.com/view/37.htm" \t "_blank)、[计算机硬件](http://baike.baidu.com/view/1024731.htm)、[外设](http://baike.baidu.com/view/1162088.htm)、网络等其他元素结合在一起，进行信息系统的各种[组装测试](http://baike.baidu.com/view/2303221.htm)和[确认测试](http://baike.baidu.com/view/106760.htm)，系统测试是针对整个产品系统进行的测试，目的是验证系统是否满足了需求规格的定义，找出与需求规格不符或与之矛盾的地方，从而提出更加完善的方案。系统测试发现问题之后要经过调试找出错误原因和位置，然后进行改正。是基于系统整体需求说明书的黑盒类测试，应覆盖系统所有联合的部件。对象不仅仅包括需测试的[软件](http://baike.baidu.com/view/37.htm)，还要包含软件所依赖的硬件、外设甚至包括某些数据、某些支持软件及其接口等。

系统测试是保证系统质量的关键，是对整个系统开发过程的最终审查。在管理信息系统开发周期的各个阶段都不可避免地会出现差错，系统开发人员应力求在每个阶段结束之前进行认真、严格的技术审查，尽可能及时发现并纠正错误，但开发过程中的阶段审查并不能发现所有的错误。这些错误如果等到系统投入运行后再纠正，将在人力、物力上造成很大的浪费，甚至导致系统的失败。此外，在程序设计过程中，也会或多或少地引入新的错误。因此，在应用系统投入之前必须纠正这些错误，这是系统能够正确、可靠运行的重要保证。统计资料表明，对于一些较大规模的系统来说，系统调试的工作量往往占据程序系统编制开发总工作量的40%以上。

很多人认为“测试是证明程序中不存在错误的过程”、“程序测试的目的是要证明程序正确地执行了预期的功能”、“程序测试的过程是使人们确信程序可完成预期要完成的工作过程”。但却是错误的定义。测试的定义应该是：为了发现错误而执行程序的过程。从这个定义出发可以看出应该把查出了新错误的测试看作是成功的测试，没有发现错误的测试则是失败的测试。

系统测试是保证系统质量和可靠性的关键步骤，是对系统开发过程中的系统分析、系统设计和系统实施的最后复查。基于以上系统测试概念和目的，在进行系统测试时应遵循以下基本原则。

(1) 测试工作应避免由原来开发软件的个人和小组承担。测试工作应由专门人员来进行，会更客观、更有效。

(2) 测试用例不仅要确定输入数据，而且要根据系统功能确定预期输出结果。将实际输出结果与预期输出结果相比较就能发现程序是否有错误。

(3) 设计测试用例不仅要包括有效合理的输入数据，也要包含不合理、无效的输入数据。

二、系统测试内容

系统的测试工作可以分为模块测试、子系统测试和系统测试。

1.模块测试

模块测试式以系统的程序模块为对象进行测试，验证模块功能及其接口与设计说明书是否一致。在一个设计系统中，每个模块完成一个清晰单一定义的功能，可以把模块作为一个单独的实体来测试。模块测试的目的式测试模块在语法、格式和逻辑上的错误，保证每个模块本身能正常运行，在该测试中发现的问题大都是程序设计错误或系统设计中的错误。

模块测试主要从下述几个方面进行。

（1）模块功能。测试模块功能是否正常，即能否在正常输入情况下获得应该得到的结果.

（2）内部数据结构。测试模块内部数据结构的内容、数据行使及数据之间的相互关系是否正确，这些数据结构是否能够正确地支持模块功能的实现。

（3）出错处理。测试模块运行出错时，出错处理功能是否能及时启动并有效工作。

（4）边界条件。软件往往容易在边界条件上发生问题，因此需要测试模块在其各种边界参数处能否正常工作。

2.子系统测试

完成每个模块的测试以后，需要按照系统设计所完成的模块结构图把它们连接成子系统，即进行子系统测试。有些模块能够单独地正常工作，但是连接起来就不能正常运行，这里可能存在接口和整体协调问题。例如数据可能在接口的传递中丢失、一个模块的运行可能会干扰其他模块运行等都是子系统测试所要关注的问题。子系统测试是在模块测试的基础上，解决模块间的相互调用问题，主要测试子系统中各个模块的外部功能，以及模块之间的接口好调用关系。

子系统测试主要有以下内容。

（1）各模块的连接是否有错误。测试模块接口是否正常，及模块调用参数的数目、顺序和类型是否正确，数据能否正确无误地流入、流出模块；能否保证数据的有效传输、数据的完整性和一致性。

（2）运行中的人机界面及各种通信接口能否满足设计要求。

（3）所涉及的外部设备是否能正确连接，与相关外部设备的通信功能是否满足设计要求。

3.系统测试

系统测试就是将经过子系统测试的模块群装配成一个完整的系统进行测试，以检查系统是否达到了系统分析的要求，系统测试的依据是系统分析报告，系统的测试不仅是对软件的测试，而且是对系统的软件与硬件一同进行测试。系统测试包含功能测试、恢复测试、安全测试、强度测试和性能测试。

（1）功能测试。系统功能测试是对系统分析报告中的目标和软件系统所实现的目标进行逐个比较，用以检查目标功能是否遗漏。

（2）恢复测试。系统恢复测试主要检查系统的容错能力。测试系统出错时，能否在制定的时间间隔内修正错误并重新启动系统。

（3）安全测试。系统安全测试主要检验系统是否存在安全保密漏洞。，在测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。

（4）强度测试。系统强度测试主要检验系统的信息处理能力最高能达到什么实际限度。强度测试迫使系统在它的设计能力极限状态下进行，进而测出系统的极限。

（5）性能测试。对于一些实时和嵌入式系统，软件部分即使满足功能要求，也未必满足性能要求。性能测试要全面、可靠地系统运行中的各种性能指标是否能够达到用户的实际需要。模块测试、子系统测试和系统测试，每一步都是在前一步的基础上进行，其过程如图所示。

模块测试

子系统测试

系统测试

【系统的测试内容】图例

6.3.2软件测试的方法

软件测试的常用方法：

|  |  |
| --- | --- |
| 测试名称 | 测试内容 |
| Black Box黑盒测试 | 把软件软件系统当做一个“黑箱”，无法了解或使用系统的内部结构及知识。从软件的行为，而不是内部结构出发来设计测试。 |
| White Box白盒测试 | 设计者可以看到软件系统的内部结构，并且使用软件的内部知识来指导测试数据及方法的选择。 |
| Gray Box灰盒测试 | 介于黑盒和白盒之间 |

1.黑盒测试：

[黑盒测试](http://baike.baidu.com/view/51274.htm)，英文是Black Box Testing。又称[功能测试](http://baike.baidu.com/view/651435.htm" \t "_blank)或者[数据驱动测试](http://baike.baidu.com/view/760744.htm)。测试者把程序看成是一个黑盒，完全不考虑程序内部结构和内部特性而进行的测试。

黑盒测试是根据[软件](http://baike.baidu.com/view/37.htm" \t "_blank)的规格对软件进行的测试，这类测试不考虑软件内部的运作原理，因此软件对用户来说就像一个黑盒子。[软件测试人员](http://baike.baidu.com/view/4005893.htm)以用户的角度，通过各种输入和观察软件的各种输出结果来发现软件存在的缺陷，而不关心程序具体如何实现的一种软件测试方法。

[黑盒测试](http://baike.baidu.com/view/51274.htm)常用工具有：[AutoRunner](http://baike.baidu.com/view/1629309.htm)、winrunner、[loadrunner](http://baike.baidu.com/view/812367.htm" \t "_blank)。

在黑盒测试中所采用的方法主要有等价类划分和边界值分析等方法。

（1）等价类划分

程序中所输入的数据按照程序功能说明可以分成若干个等价类，按输入条件可以将每一个等价类分成有效输入等价类与无效输入等价类两种。因此，对于每一个有效或无效的等价类可以设计一些测试用例，如果这些测试用例不出现错误，说明对于同一类的其他数据也不会出错，否则肯定出错。例如，在测试销售定价处理程序时，由于产品销售价格主要依赖于客户所订购的产品数量、客户是否是老客户、客户是否具有拖欠贷款的记录等条件。因此在对该程序测试中，可以按照这三个条件设计不同的等价类测试用例。

（1）边界值分析

由于许多软件在处理边界值时容易发生错误，用大于、等于、小于边界值的数据作为测试用例更容易发现程序中的错误。而等价类测试方法往往是在某一等价类中进行随机选择，未必代表了边界状况。因此，边界值测试方法可以对等价类测试方法进行有效的补充。例如，某一程序的输入数据位0～1000，可以选择-0.1、0、0.1、999.9、1000、1000.1作为测试用例。

2.白盒测试

白盒测试，英文是White Box Testing。又称结构测试或者逻辑[驱动](http://baike.baidu.com/view/43111.htm" \t "_blank)测试。是允许人们检查程序的内部结构，是测试者从检查程序的逻辑着手，得出测试数据，进行测试的过程。是把测试对象看作一个打开的盒子。利用白盒测试法进行[动态测试](http://baike.baidu.com/view/106213.htm)时，需要测试[软件](http://baike.baidu.com/view/37.htm)产品的内部结构和处理过程，不需测试软件产品的功能。

[白盒测试](http://baike.baidu.com/view/51297.htm)法的覆盖标准有[逻辑覆盖](http://baike.baidu.com/view/178784.htm)、循环覆盖和基本[路径测试](http://baike.baidu.com/view/4533384.htm)。其中[逻辑覆盖](http://baike.baidu.com/view/178784.htm)包括[语句覆盖](http://baike.baidu.com/view/1741053.htm)、[判定覆盖](http://baike.baidu.com/view/953374.htm)、[条件覆盖](http://baike.baidu.com/view/1289596.htm)、判定/条件覆盖、[条件组合覆盖](http://baike.baidu.com/view/229032.htm" \t "_blank)和[路径覆盖](http://baike.baidu.com/view/2193850.htm)。它知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构[测试程序](http://baike.baidu.com/view/420822.htm)，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于[软件](http://baike.baidu.com/view/37.htm)验证。

[白盒测试](http://baike.baidu.com/view/51297.htm)常用工具有：Jtest、VcSmith、Jcontract、C++ Test、CodeWizard、logiscope。

\*总结：   实际工作中，对系统的了解越多越好。目前大多数的测试人员都是做黑盒测试，很少有做白盒测试的。 因为白盒测试对软件测试人员的要求非常高，需要有很多编程经验。做.NET程序的白盒测试你要能看得懂.NET代码。做JAVA程序的测试，需要你能看懂JAVA的代码。 如果你都能看懂了，你还会做测试么？

* + 1. 软件测试步骤

一、软件测试的过程

系统测试是开发过程中一个独立而且非常重要的阶段，也是保证开发质量的重要手段之一。测试过程基本上与开发过程平行进行。在测试过程中，需要对整个测试过程进行有效的管理，保证测试的质量和效率。一个规范的测试过程通常包括以下基本的测试活动。

1. 制定测试计划

在制定测试计划时，需要充分考虑整个项目的开发时间和开发进度，以及一些人为因素和客观条件等，使得测试计划是可行的。测试计划的内容主要有：测试的内容、进度安排、测试所需的环境和条件、测试培训安排等。

1. 编制测试大纲

测试大纲是测试的依据，它明确详尽地规定了在测试中针对系统的每一项功能或特性所必须完成的基本测试项目和测试完成的标准。

1. 设计和生成测试用例

依据测试大纲设计和生成测试用例，产生测试设计说明文档，其主要内容有被测项目、输入数据、测试过程和预期输出结果等。

1. 实施测试

测试的实施阶段是由一系列的测试周期组成的。在每个测试周期中，测试人员和开发人员将依据预先编制好的测试大纲及准备好的测试用例，对被测系统或设备进行完整的测试。

1. 生成测试报告

测试完成后，要形成相应的测试报告。主要对测试进行概要说明，列出测试的结论，指出错误和缺陷。另外，给出一些建议。

二、软件测试的方法

对软件进行测试的主要方法有人工测试和机器测试。人工测试采用人工方法进行，目的在于检查程序的静态结构，找出编译不能发现的错误。经验表面，良好的人工测试可以发现程序中30%～70%的编码和逻辑设计错误，从而可以减少机器测试的负担。机器测试是将事先设计好的测试用例作用于被测试程序，对比测试结果和预期结果的差别以发现错误。机器测试只能发现错误的症状，不能进行问题定位，而人工测试一旦发现错误，就能确定问题的位置、类型和性质。对于一些类型的错误，机器测试比人工测试有效，但对另一些类型的错误则人工测试更有效。因此，应根据实际情况来选择测试方法。

测试模型：

程序设计

系统设计阶段

系统分析阶段

模块测试

高级测试

应用需求

系统外部规范设计

逻辑结构设计

程序结构设计

编写代码

子模块测试

调试代码

功能测试

模块整体设置

验收测试

安装测试

系统测试

1. 系统测试步骤

合理安排测试步骤对于提高测试效率、降低测试成本有很大的作用。管理信息系统的测试步骤是：分别按硬件系统、网络系统和软件系统进行测试，最后对整个系统进行总的综合测试。测试的步骤如下：

（1）硬件测试

在开发建设管理信息系统时，通常需要根据项目情况选购硬件设备。在硬件设备采购到货后，应在各相关厂商的配合下进行初始验收测试，初验通过后，再将其与软件、网络等一起进行系统测试。初验测试所做的主要工作有一下几点。

1. 配置检测。检测是否按照合同提供了相应的配置，如系统软件、硬盘、内存、CPU等的配置情况。
2. 硬件设备的外观检查。所有设备及配件开箱后外观有无明显的划痕和损伤，这些包括计算机主机、工作站、磁带库、磁盘机柜和存储设备等。
3. 硬件测试。首先进行加电检测，查看运行状态是否正常，有无报警、屏幕有无乱码提示和死机现象，以及是否能进入正常提示状态；然后进行操作检测，用一些常用的命令来检测机器是否能执行命令，执行结果是否正常，如文件复制、显示文件内容、建立目录等，最后检查是否提供了相关工具，如帮助系统、系统管理工具等。

通过以上测试，要求形成相应的硬件测试报告，报告中应包含测试步骤、测试过程和测试结论等。

（2）网络测试

如果开发的管理信息系统需要在局域网或广域网上运行，通常需要选购网络设备。在网络设备按合同到货后，应在各相关厂商的配合下进行初始验收测试，初验通过后，再将其与软件、网络等一起进行系统测试。初验测试所做的主要工作有一下几点。

1．网络设备的外观检查，所有设备及配件开箱后外观有无明显的划痕和损伤，这些包括交换机、路由器等。

2．硬件测试。对交换机、路由器等进行加电检测，观看运行状态是否正常，有无错误、有无报警等。

3.网络连通测试。检测网络是否连通，检测服务器和各工作站是否能通过网络设备建立正常连接。

通过以上测试，要求形成相应的网络测试报告，报告中应包含测试步骤、测试过程和测试结论等。

（3）软件测试

软件测试实际上分为4个步骤：单元测试、集成测试、确认测试和系统测试。如图所示：

硬件设备

确认后的硬件系统

模块

模块

模块

设计信息

用户信息

可以运行的系统

已经确认的应用软件

需求信息

已经集成的子系统

已经测试过的模块

网络设备

确认后的网络系统

图：管理信息系统测试过程

(a)单元测试。单元测试也称为模块测试。对源程序中的每个程序单元进行测试，以验证每个模块是否满足系统说明书的要求。一般情况下是编好一个程序模块就测试一个，通常采用白盒测试方法，再辅之以黑盒测试。

(b)集成测试。集成测试又称组装测试。是把模块按系统设计说明书的要求组装起来进行测试。集成测试的主要目标是发现与接口有关的问题，如数据通过接口时可能丢失，一个模块对另一个模块可能造成有害影响，将它们组合起来后可能不能实现预期的功能，全局数据结构会出现问题，误差累积到超过可接受的程度等。集成测试通常有两种方法：一种是非增量式集成测试，即先分别测试各个模块，再把这些模块组合起来进行整体测试；另一种是增量集成测试，即把下一个要测试的模块组合到已测试好的模块中，测试完后再将下一个需要测试的模块组合起来进行测试，逐步把所有的模块组合在一起，并完成测试。集成方式可采用自顶向下集成或自底向上集成。

(C)确认测试。确认测试是按照系统说明书中定义的全部功能和性能要求及确认测试计划，来测试整个软件系统是否达到了用户要求，确认测试首先要进行有效性测试及软件配置审查，然后进行验收测试和安装测试，经过管理部门的认可和专家的鉴定后，软件即可以交给用户使用。确认测试通常采用黑盒测试法。

(d)系统测试。系统测试是将已经确认的软件、计算机硬件、外设和网络等系统的各个部分连接到一起，对整个系统进行总的功能、性能等方面的测试。其目的是通过与系统的需求相比较，发现所开发的系统与用户需求不符或矛盾的地方。

6.3.4软件正确性证明

测试正确性证明是显示产品正确的一种数学技术。换句话说，产品满足规格说明。该技术有时称为验证，然而验证一次常用于表示所有的非执行测试技术，不只是正确性证明。为明确起见，将这个数学称为正确性证明，它是一个数学证明过程。

【例】为了明白如何证明正确性，考虑图a所示代码段。与代码等效的流程图（图b）所示。我们现在来证明该代码段是正确的。执行该代码之后，变量s的值等于数组y的n个元素的和。在图c中，在每个语句之前和之后，在标注字母A~H的地方，放置一个断言，也就是在每个拥有某个数字属性的地方做一个声明。我们需要做的是证明每个断言的正确性。

Int k,s;

Int y[n];

K=0;

S=0;

While(k<n)

{

S=s+y[k];

K=k+1;

}

图a：要证明是正确的代码段

K←0

s←0

s←s+y[k]

k←k+1

Is

K≥n？

图b：图a的流程图

B

YES

H

K←0

k←k+1

s←s+y[k]

s←0

Is k≥n？

C

D

E

F

G

A

NO

n∈{1,2,3 …} (输入规格说明)

K=0

K=0且s=0

K≤n且s=y[0]+y[1]+…+y[k-1]

K<n且s=y[0]+y[1]+…+y[k-1]

K<n且s=y[0]+y[1]+…+y[k]

K≤n且s=y[0]+y[1]+…+y[k-1]

K=n且s=y[0]+y[1]+…+y[n-1]

图c：带有输入规格说明、输出规格说明、循环不变式和断言的图b

输入规格说明——代码执行前在A处具有的条件是，变量n是一个正整数，也就是：

A：n∈{1,2,3。…} （1）

明显的输出规格说明是，如果控制达到H点，s的值等于储存在数组y中的n各值的和，也就是：

H：s=y[0]+y[1]+...+y[n-1] （2）

事实上，对于较强的输出规格说明，可以证明改代码段是正确的：

H：k=n且s=y[0]+y[1]+…+y[n-1] (3)

除了输入和输出规格说明，本证明过程的第三个方面是提供一个循环不变式，也就是必须在D点提供一个数学表达式，不管该循环执行了0次、1次还是许多次。要证明持有的循环不变试试：

D：k≤n且s=y[0]+y[1]+…+y[k-1] (4)

下面将表明，如果在A点输入规格说明式（1）成立，那么输出规格说明式（3）将在H点成立，也就是证明该代码段是正确的。

首先，执行赋值语句k←0，现在控制在B点，如下的断言成立：

B：k=0 （5）

为了更准确，在B点断言应该读做k=0且n∈{1,2,3…}。然而在流程图中的所有点输入规格说明式（1）都成立。为简便起见，以下“n∈{1,2,3…}”省略。

在C点，作为第二赋值语句s←0的结果，下面断言是真的：

C：k=0且s=0 （6）

现在进入循环，这里将通过推到证明循环不变式（4）确实是正确的。在该循环第一次执行之前，断言式（6）成立，也就是k=0且s=0。现在看循环不变式（4）。因为断言式（6）可以确定k=0，而且从出入规格式说明（1）可以确定n≥1，如果要求的那样，k≤n成立。进一步地说，因为k=0，k-1=-1，所以式（4）中的和是空的，而且要求s=0.循环不变式（4）因此在第一次进入循环之前是真的。

现在进行归纳假设步骤。假设在改代码段执行期间的某个阶段，改循环不变式有效，也就是k等于某个值k0。0≤k0≤n，执行到D点，有下面的断言：

D：k0≤n且s=y[0]+…+ y[k0-1] (7)

控制现在经过测试框。如果k0≥n，因为假设k0≤n,所以k0=n。按照归纳假设式（7），这意味着：

H：k0=n且s= y[0]+ y[1]+…+ y[n-1] （8）

这正好是输出格式说明式（3）。

另一方面，如果测试k0≥n？结果为否，那么控制从D点转向E点。因为k0不大于或等于n，k＜n则式（7）变成：

E：k0＜且s= y[0]+ y[1]+…+ y[k0-1] （9）

现在执行语句s ←s+y[k0],因此由于断言式（9），现在F点一定有下列的断言：

F：k0＜n且s= y[0]+ y[1]+…+ y[k0-1]+y[k0]

= y[0]+ y[1]+…+ y[k0] (10)

下一个要执行的语句是k0 ←k0+1，为了看到这个语句的结果，假定执行该语句前k0的值是17，那么式（10）中和的最后一项是y[17]。现在k0值增加1意味着，如果G点不等式成立，那么k0≤n。这样，k0增加1的结果是在G点下面的断言成立：

G: k0≤n且s= y[0]+ y[1]+…+ y[k0-1] （11）

在G点的断言式（11）与在D点假定的断言式（7）相同，但D在拓扑上与G点相同。换句话说对于k=k0，如果在D点式（7）成立，那么对于k=k0+1，它仍在D点成立。前面已经显示了k=0时循环不变式，经过推到，对于所有的k值，0≤k≤n，循环不变式（4）成立。

剩下的就是证明循环终止。最初由断言（6），k的值等于0。循环每次迭代时执行k ←k+1，使k值增加1。最后，k一定达到n值，那时将退出循环，并且断言（8）给出的s的值，这样就满足于输出规格说明式（3）。

回顾给定输入规格说明式（1），可以证明不论该循环执行0次、1次或更多次，循环不变式(4)成立。进一步地，可以证明经过n次迭代后，循环终止，而且这时k和s的值满足输出规格说明式（3）。换句话说，图4的代码经过数学证明是正确的。