什么是软件测试 软件测试就是用来确定代码完成其应该完成的，未完成其不应该完成的过程。

目的 在于发现错误，也是对软件质量进行度量和评估，以提高软件的质量。为软件可靠性提供依据，树立人们使用软件的信心。

在正确的时间用正确的方法把一个工作做正确。符合一些应用标准的要求，比如项目中的可维护性、可测试性等要求。质量本身就是软件达到了最开始所设定的要求，也代表着符合客户的需要。——质量

原理：所有测试都应追溯到用户需求，帕累托法则使用与软件测试，测试计划的制订应先于测试的执行，软件测试应从“小规模”开始，然后扩展到“大规模”，完全测试是不可能的，要使测试更为有效，测试应有独立的第三方进行。

软件测试过程：

拟定测试计划，编制测试大纲，生成测试用例，实施测试，分析测试结果

什么是测试用例？一个文档，详细说明输入、期望输出，和为一测试项所准备的一组的执行条件

白盒测试 基本路径测试 条件测试、数据流测试

黑盒测试 ：边界值分析。因果测试。等价划分。、

测试的类型

按开发阶段有单元、集成、确认、系统、验收 测试

测试技术划分： 白盒，黑盒，灰盒

执行状态：静态（不运行程序，人工或利用工具对文档进行检查）、动态（利用工具或人工）

静态：走查审查 符号执行 动态：单元测试、集成测试、确认测试、系统测试、验收测试、白盒测试、黑盒测试、及灰盒测试等。

还有：开发方、用户测试、第三方

需求分析和定义——系统设计——详细功能设计——编码 =构建过程

验收测试————系统测试——功能测试————单元测试=验证过程 //V模型

测试用例应当由测试输入数据和与之对应的预期输出结果两部分组成。

避免检查自己的程序，应当包括合理的和不合理的输入条件，尽早地和不断地

该具备的态度应具备的素质

三个数值，代表三条边，判断属 不规则普通三角形，等腰三角形还是属等边三角形

判断程序是否正确。

技术性，态度更重要，心理学，测试是为了发现错误而执行程序的过程，暗示了是一个破坏性的过程。成功的测试，发现了某个新错误的测试。

不成功的测试：只未能适当地对程序进行检查，在大多数情况下，未能找出错误地测试被认为是不成功的。

一个成功地测试用例，通过诱发程序发生错误，可以在这个方向上促进软件质量地改进

最终，通过软件测试建立某种程度的信心：软件做了其应该做的，未做其不应该做的

经济学 发现程序不按其规范正确运行的环境条件

穷举输入测试 这是无法实现的 要考虑经济问题，是不可能实现的，花费很多精力，很多时间。白盒测试：逻辑驱动测试，忽略了程序的规范，检查逻辑与内部 穷举路径测试：所有可能的控制流路径

程序中的路径可能达到天文数字，虽然测试了很多路径，但还是无法保证程序是正确的1设计规范2缺少路径3不会暴露数据敏感错误。

穷举测试强于穷举路径

原则：必须要有预期输出或结果的定义，程序员避免自己编写的程序，编写软件的组织不应当测试自己编写的软件，应当彻底检查每个测试的执行结果，根据有效和预期的输入情况，还有根据无效和未预料到的输入情况。

检查程序是否未做其应该做的仅是测试的一半，测试的另一半是检查程序是否做了其不应该做的。应避免测试用例用后即弃，除非软件本身就是一个一次性的软件。8，不应默许假定不会发生错误9，某部分存在更多错误的可能性与该部分已发现的错误成正比。10，软件测试是一项极富创造性、极具智力挑战性的工作。

重要的测试原则：为发现错误而执行程序的过程；尽量避免自己的程序：好的测试用例能够对未发现的错误高度敏感；成功的测试用例能发现未知的错误，成功的测试用例需要仔细定义输入输出的期望值，成功的测试用例需要仔细研究分析测试结果 作业：

心理学 为何不可行 穷举路径和输入，基本原则是哪些

代码的检查与走查，代码检查，用于代码检查的错误列表，代码走查，桌面检查。同行评审。

软件测试是一项技术性工作。软件测试的唯一方法是在计算机上运行吗？？？

人工测试：非基于计算机测试的过程

程序开始编码之后，基于计算机的测试之前使用人工测试。、从两个方面提高了测试的功效和可靠性，错误发现得越早，改正错误的成本越低，正确改正错误的可能性也越高。要经历一个心理上的改变，改正某个由于计算机测试发现的错误所犯的失误较之前更多些。人工测试的3种主要方法：

代码检查。代码走查。可用性测试。

、要求组成一个小组来阅读或直观检查特定的程序，头脑风暴会的目标是找出错误来，不必找出改正错误的方法。有其他人参与，精确定位，若发现成批的错误可一同修正，30%-70%的逻辑设计和编码，与计算机测试形成互补。修改比编写更容易产生错误，修改后仍需人工测试。小组：4人，协调员分发，作者，其它专家高效，一个或几个子模块 对事不对人，态度，找出错误，提升质量，风格和凝聚力。

错误检查不能模糊，错误列表独立于编程语言。数据引用错误，数据申明错误，运算错误（除数为0？）（基于二进制，运算结果是否不精确？）（优先顺序）（整数运算使用不当），比较错误（内容是否正确）（是否有用二进制表示的小数或浮点数的比较运算）（例：i>x||y a==b==c x>y>z:x>y&&x>z if(x==0&&(y/x)>z)，控制流错误(包含多条分支路径索隐变量的值是否会大于可能的分支数量，是否所有循环最终都终止了)（程序，模块或子程序是否最终都终止了）（仅差一个的错误）（是否存在不能穷尽的判断？），接口错误（形参，实参的数量，量纲，数据类型或大小，内置函数，全局变量）（是否以实参形式传递过 CALL SUBX（J,3）——语句很危险），输入/输出错误（I/O，file not found错误），其他检查（是否遗漏了某个功能？）（是否具有足够的鲁棒性）。

代码走查，以小组为单位，3-5人，协调员，秘书，测试人员，极富经验的程序员，）使用了“计算机”，被指定测试人员会准备测试用例参加会议，参与者提出的建议应该是针对程序本身。作用：发现易出错的程序区域，通过接触软件错误、编程风格和方法获得教育。

桌面检查：古老的人工检查方法，个人进行的，桌面检查胜过没有检查，但是逊色于代码走查和代码检查。同行评审，自我评价的手段，易于理解？高层次的设计是否合理且可见，低层次的设计是否合理且可见，总的评价和建议。

大多数软件都应使用人工测试：

利用错误列表进行代码检查，小组代码走查，桌面检查，同行评审，可用性测试。

a>0and b>0? C=c/a: (a>1 or c>1 ?c=c+1:c = b+c)

圈复杂度=流图中空间区域的数量=流图中边-节点+2=流图中判定节点的个数+1

Do while

If … then

…

Else if … then

…

Else

…

End if

End if

End do

End

1-11

1-2-3-4-5-10-1-11

1-2-3-6-8-9-10-1-11

1-2-3-6-7-9-10-1-11

[1-2(3)-4(5)-10-1-2(3)-6-7-9-10-1-11]

1 绘制流图2，确定圈复杂度，3，确定线性独立路径的基本路径集合，4，设计测试用例覆盖每条基本路径。

白盒测试：

白盒测试、基本路径测试、条件测试、数据流测试、循环测试。白盒测试又称为玻璃和测试，是一种基于源程序或代码的测试方法，分为静态和动态两种类型。生成测试用例并驱动被测程序，代码检查法，功能测试，数据驱动测试，黑盒测试，结构测试，逻辑驱动测试，白盒测试，静态的和动态的，走读，运行程序

自动测试和手工测试 rational test 手工模拟用户操作

白盒测试方法：语句覆盖，判定覆盖，条件覆盖，判定条件覆盖，条件组合覆盖，路径覆盖，基本路径测试法。

白盒测试方法：逻辑覆盖、基本路径测试、数据流测试

逻辑覆盖分为判定覆盖，条件覆盖，判定-条件覆盖，条件组合覆盖，

基本路径测试：设计测试用例

数据流测试：关注变量的赋值和使用，基于变量的定义-使用链导出程序的测试路径，设计测试用例。

每个可执行语句至少被执行一次，

如果是顺序结构，就是让测试从头执行到尾

如果有分支、条件和循环，需要执行足够的测试用例覆盖全部语句

简单条件，组合条件，关系表达式，判定条件覆盖，组合条件C为真或假的分支以及每个条件的可能取值至少一次。条件组合，让每个条件出现真假的结果的所有可能组合至少出现一次。每个判断的判定结果也至少出现一次。T1T2 T1F2 F1T2 F1F2 T3T4 T3F4 F3T4 F3F4 判定取值。A>0&&b>0 a>1||c>1.输入

覆盖条件

a=2,b=1,c=2

a=2, b=-1, c=-1

a=-1, b=1, c=2

a=-1, b=-1, c=-1 未覆盖所有路径，故基本路径测试法

下面的测试用例覆盖了哪些方法：

绘制流图

确定圈复杂度

确定线性独立路径的基本集合

设计测试用例覆盖每条基本路径。、

程序流程图，框图，起始框，终止框，执行框，判断框。

流图：每个圆表示一个或多个非分支PDL（program design language）或源代码语句

流图描述的是逻辑控制流，箭头称为边或连接，表示控制流。

1-2（3）-4（5）-10-1-11，1-11，1-2（3）-6-8-9-10-1-11，1-2（3）-6-7-9-10-1-11

程序设计中遇到复合条件时，生成的流图变得稍微复杂。

a

b：假 x:真

y：假 x：真

输出：所有路径的出口。

独立路径也称基本路径，是指在程序入口和出口之间的任一路径，其间不存在两条长度大于2的相同子路径。

有的书中定义：独立路径是指程序中至少引进一个新的处理语句集合或一个新条件的任一路径。采用流图的术语，即独立路径必须至少包含一条在定义路径之前未被用到的边

流图定义：

G=(V, E)，其中V是顶点的集合，E是有向边的集合。

D是原子二元判定条件的节点集。

是一条基本路径，

如何才能知道需要寻找多少条路径呢？圈复杂度

圈复杂度以图论为基础，为我们提供了非常有效的软件度量。

圈复杂度（cyclomatic complexity）

圈复杂度是一种为程序逻辑复杂性提供定量测度的软件度量。当该度量用于基本路径测试方法，计算所得的值给出了程序基本集的独立路径数量，这是为确保所有语句至少执行一次而必须进行测试数量的下界。

三种方法计算圈复杂度，给定流图G的环形复杂性——V(G)，V(G)=流图中区域的数量。

给定流图G, V（G）=E-N+2。V(G)=P+1。、

复杂度越高，出错的概率越大。

V(G)=区域数量（由节点、连线包围的区域，包括图形外部区域）V(G)=边数量-节点数量+2

V(G)=判断节点数目+1

源代码：

do while records remain

read record;

if record field 1=0 then

store in buffer;

increment counter;

else if record field 2=0 then

reset counter;

else store in file;

end if

end if

end do

end

开始节点与结束节点，判断节点+箭头，输入框，执行框是矩形。

流程图简化2,3 4,5

V(G)=4，1-2（3）-11, 1-2(3)-4(5)-10-1-11

1-2(3)-6-7-9-10-1-11

1-2(3)-6-8-9-10-1-11

1-2(3)-4(5)-10-1-2(3)-6-7-9-10-1-11

region1 region2 region3 region4 V(G)=4 确定独立路径的路径集合

基本集：由独立路径构成的集合。

由基本集导出的测试用例，保证每行代码语句至少被执行一次

基本集合不一定唯一

测试用例覆盖集合中每条路径 绘制程序流图，并转换为流图，最好每个单元都进行基本路径测试，对关键组件则是必要的。

基本路径测试并不是测试所有路径的组合，仅仅保证每条基本路径被执行一次。

补充：计算V(G)并不一定要画出流图，计算PDL中的所有原子条件语句数量（while及if条件语句），然后加1即可得到环形复杂性。

应该注意到，如果是复合条件语句，需要求出原子条件语句数量。

测试用例导出步骤

利用基本路径测试产生测试用例的一系列步骤

以设计或代码为基础，画出相应的流图

确定结果流图的环形复杂性

确定线性独立的路径的一个基本集

准备测试用例，强制执行基本集中每条路径

示例二

4.准备测试用例，强制执行基本集中每条路径，测试人员可选择数据以便在测试每条路径时适当设置判定节点的条件。

路径1测试用例：

输入条件：L=()，即列表为空

期望结果：L=()，无列表清理处理

路径2测试用例：

输入条件：L=(Ap)

期望结果：L=(Ap)

路径3测试用例：

输入条件：L=(Ap Aq )且Ap=Aq

期望结果：L=(Ap)，从列表中清理Aq

路径4测试用例：

输入条件：L=(Ap Aq)且Ap<>Aq

期望结果：L=(Ap Aq)

例子3

书面作业，a>0and b>0

a>1or c>1

图矩阵法

一种称为图矩阵的数据结构很有用

图矩阵是一个正方形 行列数=流图的节点数

每列和每行都对应于标识的节点，矩阵项对应于节点间的连接（边）

数据流测试 数据流测试方法按照程序中的变量定义和使用的位置来选择程序的测试路径。

已经有不少关于数据流测试策略的研究

为了说明数据流测试方法，假设程序的每条语句都赋予了唯一的语句号，而且每个函数都不改变其参数和全局变量。

数据流测试 def(S)={x|语句S包含X的定义}

use(S)={x| 语句S包含X的使用}

def(1)={x}

use(1)={y,z}

use(2)={z}

use(3)={y}

def(4)={w}

use(4)={x}

如果存在从S到S‘的路径，并且该路径不含X的其他定义，则称变量X在语句S处的定义在语句S’仍有效或称为定义清纯（def-clear）

数据流测试

如果存在从S到S‘的路径，并且该路径不含X的其他定义，则称变量X在语句S处的定义在语句S’仍有效或称为定义清纯(def-clear)

变量x的“定义使用关联”(du-association)形式如[x,S,S’]，其中S和S‘是语句号，X在def(S)和use(S’)中，而且语句S定义的X在语句S’有效。

图中变量x的du-关联为[x,1,4]

数据流测试

一个变量的使用可以是计算使用（c-use）或断言使用（p-use）。

一个变量的du-链是一条路径，这条路径是从变量的定义到变量的使用之间无任何重定义（也就是说，definition-clear）。

对于c-use来说，du-链是从含有定义语句到含有计算使用语句之间的路径。

对于p-use来说，du-链是从含有定义语句到包含两个执行分支断言使用语句之间的路径。

但是，数据流测试的覆盖率度量和路径选择比条件测试更为困难

循环测试-1

目标：在循环内部边界上执行测试

1. 简单循环（迭代次数n）

完全跳过循环，只经过循环一次，经过循环两次，经过循环m(m<n)次，分别经过循环n-1,n,n+1次

1. 嵌套（neated）循环

从最内层循环开始，将其他循环设置成最小值。

对最内层循环使用简单循环测试，而使外层循环的迭代参数（即循环计数）最小，并增加其他的测试用例来测试范围外或排除的值进行。

由内向外构造下一个循环的测试，但其他的外层循环为最小值，并使其他的嵌套循环为“典型”值

继续直到测试完所有的循环。

1. 串行连接的循环

独立循环->可以分别看着简单循环测试

依赖性循环->可以看着是嵌套循环

1. 不规则循环

在尽可能地情况下，要将这类循环重新设计为结构化地程序结构，redo!

总结

白盒测试是一种基于源程序或代码地测试方法，分为静态和动态两种类型。

静态方法是指按一定步骤直接检查源代码来发现错误，而不用生成测试用例并驱动被测程序来发现错误，也称为代码检查法；

动态方法是指按一定步骤生成测试用例并驱动被测程序运行来发现错误。

静态方法有桌面检查、代码审查及走查；动态方法有基本路径测试、条件测试、数据流测试及循环测试。

白盒测试是一种主要地单元测试方法，一般由软件开发人员进行。

白盒测试过程主要有这样五个步骤：根据源程序画程序图，生成测试用例，执行测试，分析覆盖标准，判定测试结果。

第四章 测试用例地设计

黑盒测试

等价类划分法

边界值分析法，判定表法，因果图法，正交试验法，功能图法，错误推测法

又叫功能测试。

功能性需求，生成输入条件集来检测所有功能需求。

黑盒测试并不是白盒测试的替代品，而是配合白盒测试发现其他类型的错误。

黑盒测试试图发现的错误类型：

功能不对或遗漏，界面错误，数据结构或外部数据结构访问错误，

性能错误，初始化错误，终止错误

前置条件：测试执行所必须满足的条件

后置条件：发生这个用例之后的结果，会产生的影响

等价类关系

自反性，对称性，传递性，

等价类划分法试图生成那种可以揭露一类错误的测试用例，从而减少必须生成的测试用例的总数。

等价类划分法的测试用例设计是基于对输入条件的等价类评估。

将程序可能的输入数据分成若干个子集，从每个子集选取一个代表性的数据作为测试用例，等价类是某个输入域的子集，在该子集中每个输入数据的作用是等效的

分为有效等价类和无效等价类。

有效等价类是有意义的、合理的输入数据，可检查程序是否实现了规格说明中所规定的功能和性能。

无效等价类是不满足程序输入要求或者无效的输入数据构成的集合。使用无效等价类，可以测试程序/系统的兼容性-对异常情况的输入处理。

在需求规格说明书的基础上划分等价类，列出等价类表

软件不仅要接受合理的数据，也要能经受意外的考验。经过正反的测试才能确保软件具有更高的可靠性。

负整数 0~100间的整数、大于100的整数

期望输入值是10，则有效等价类为{10}，无效等价类为所有小于10的整数和所有大于10的整数。

如果输入条件是一个集合的成员，如期望输入条件是“sunos”，则有效等价类为{“sunos”}，无效等价类为所有的集合：-{“sunos”}

等价类指南，如果输入条件是一个布尔量，可以确立一个有效等价类和一个无效等价类。

例如：

允许的最低GRE成绩，电话号码可以根据区号来划分等价类，装船出货可以根据服务和区域来划分等价类。

确立等价类的方法（1）在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。less than range .in range .greater than range; less than value.value .greater than value.

特殊数值：

* 1. 一个极大的数，产生输出的为零的数值，空数组，空列表，空栈，满的数组。

测试用例可以合并，唯一，当且仅当一个成立，没有00，11. 互斥，可以有00.

不可以组合，不能找到所有组合，。是可以合并的，结果相同，由一个原因决定，与N的取值没有关系。只有两种组合。资源有限的情况下。正交数组表，例子，同一时间只改变一个输入项，功能图法，动态说明：迁移。难度。流图能压缩一定要压缩，全定义，定义了，就du链，形式一样，测一次，测中间值，结束，问题，最后一道题目来自实验报告中的题目，实验报告的题目映射到考试，实验报告。静态说明，动态说明，箭头，状态迁移图，条件，从一个状态到另一个状态的改变，复杂，示例，场景，两种方法，路径、分支，条件；因果图法，功能图测试用例，

状态图分析，金额，国家，清除，退出，报错，输入事件列举，国家未选择，人民币未输入。1.列举所有的输入，输入代表迁移，动态的，变化，所有的迁移，2.开始画，空闲状态，进行输入，产生新的状态，3.分析新产生的状态，输入，产生新的状态，重复2.3步骤，直到不能够产生新的状态为止。功能分析法，不能一对一的看的，只有用状态图，银行，随输入改变，迁移，每一种状态，转换，最不常用的分支，导致的不合理错误状态，有容错，自动化的工具，黑盒多，全 好，软件状态是核心，输入易获得，第一步所有的，进行的输入，看得到的，引申，为止。课后题目。不好解决，一种方法，很多方法，针对很多输入，很多状态，留给同学们的。上下键，有数字键，开门键，关门键，电断了。从开机运行开始分析，这个作业，发布。白盒插桩，在程序内写代码，拔掉，认识，桩要拔掉，支持黑盒测试的，独立于代码的，有两种，测试语言。语言，程序，特定的输出，给输入，有输出，预期的输出。2.随机数据生成器，这个根据输入，随机，好。测试预言，条件，谓词。喂鱼。白盒写代码，黑盒测试规格说明书，条件，喂鱼，时间间隔，半小时，改成1秒，10秒，不允许修改源代码，系统时间，安排表。S0，S出发，

先划分，取边界值，子域里取随机值，public static double random() 0.0<=math.random()<1.0

(int)(Math.random()\*100)+1

测试降低风险，要求高，优先级别，评级，低优先级的，高的先，影响程度和概率，步骤，列出功能和特性，可能性，评估程度，编写测试方案，错别字低风险，不能语音聊天，发送不出去，语音不能转换为文字。主动，发送，结果，被动，监控，白+黑，总结，内部逻辑，合法输入，输出，规约，彼此独立，偏好，功能，条件集，不是替代品，放到一起，强大，独立的，互为补充，互相独立的，微弱，错误推测法，经验非常宝贵。边界值，6位的数字，长度，类型，范围，正常的1个，长度短一点，000000，-1，-2，数字，0，空，输入小星星，特殊值，对比一下。1024，0 -48，9-57，A-65，a-97，编辑域，任选，都不，随机，有些选，特殊的值，空，等价类划分，一个发现一类的错误，输入条件分析，若干个子集，等效的，有效等价类和无效等价类。建立等价类表，有效无效，类型，唯一的编号，重要，很多种，非常重要，好的方法，设计新的用例，覆盖尚未覆盖的等价类，遵循覆盖更多的，一个覆盖多个，最好不过，尽可能多的，后续例子，重复c步骤，将所有等价类，设计新的用例，只覆盖一个无效等价类，不一样，50000，一对一的设计，两个对应两个，都被覆盖。6个步骤，分情况讨论，检审走，规范性，逻辑性，经验，逻辑性，结构，运行测试用例，发现程序中的错误而运行程序的过程，输入加预期结果，比较，设计测试用例，两大类型，白盒很多准则，基本路径测试法，把准则中要覆盖的基本路径领出来，运行它，得到结果，流程，设计测试用例，确定测试覆盖的准则，对应要覆盖的内容，设计，覆盖准则，运行它，有效等价类，区别，无效等价类，一对一的，边界值，输出为0，空的，边界值，离不开等价类的划分。大于，小于，大一点点，小，等于，

失败，代码有问题，错误，测试本身或环境有问题. public static Test suite(){ return new TestSuite(SimpleTest.class) } suite.addTest(new TestSuite(“testMax”),小单元到大单元。性能。