黑盒测试

1、黑盒测试的定义

黑盒测试又叫做功能测试或数据驱动测试，它是把测试对象看做一个黑盒子，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。

注重于程序的外部结构，主要对软件功能要求、软件界面、外部数据库访问及软件初始化等方面进行测试。测试者只要从程序接口处进行测试，以程序需求说明为测试依据，测试程序是否满足用户的需求，因此是从用户观点出发的测试。

2、黑盒测试主要发现的错误类型

（1）检测功能是否有遗漏；
（2）检测性能是否满足要求；
（3）检测人机交互是否有错误；
（4）检测界面是否有错误；
（5）检测数据结构或外部数据库访问是否有错误；
（6）检测接收数据和结果输出是否错误；
（7）检测程序初始化和终止方面是否有错误。

3、黑盒测试的方法

根据一些相关条件和方法对较典型的测试用例进行测试，来发现软件中存在的缺陷。现在大多数测试生成方法，都是通过选取软件输入域的一个子集作为测试集来测试软件的。

黑盒测试常用的方法和技术有：**等价类划分法、边界值分析法、决策表法、因果图法**等。

（1） 等价类划分方法

概念：把所有可能的输入数据，即程序的输入与划分成若干部分，然后从每一部分中选取少数有代表性的数据作为测试用例。使用这一方法设计测试用例要经历**划分等价类**（列出等价类表）和**选取测试用例**两步。

缺陷的定位：一个软件的全部输入的集合可以至少分为两个子集：一个包含所有正常和合法的输入；另一个包含所有异常和非法的输入。对于这两个子集又可以进一步划分为若干子集，以便软件针对不同的子集，其运行的结果不同。

等价类划分方法就是要从这两个集合或其子集中选择适当的输入作为测试用例，以便发现软件中存在的缺陷。

等价类的划分：等价类划分的**原则**是用同一等价类中的任意输入对软件进行测试，软件都输出的相同的结果。全部等价类的测试用例就构成了完整的测试用例集。对于同一输入域进行等价类划分，其结果可能不唯一的。因此，利用等价类划分的方法产生的测试用例集也可能不同。所以测试用例集的故障检测效率往往取决于测试人员的测试设计的经验、对软件需求的熟悉程序等。

① 划分等价类

在划分等价类时，分为有效等价类和无效等价类。

有效等价类是指符合程序规格说明书，有意义的、合理的输入数据所构成的集合。有效等价类可以是一个，也可以是多个。利用有效等价类，可以检查软件功能和性能是否符合规格说明书中的要求。

无效等价类是指不符合程序规格说明书、不合理或无意义的输入数据所构成的集合。可以是一个，也可以是多个。利用无效等价类，可以检查软件

功能和性能的实现是否有不符合规格说明书的地方。

② **常用的等价类划分原则**

i.变量的等价类划分

取值范围：如果输入条件规定了一个取值范围或值的个数，则可以定义一个有效等价类和两个无效等价类

字符串：至少分为一个包含所有合法字符串的有效等价类和一个包含所有非法字符串的无效等价类。

枚举变量：每个取值对应一个有效等价。针对枚举类型，对于某些特定的取值范围，有可能无法确定非法测试输入值。对于布尔变量，只有两个合法取值（真值与假值）

数组：数组是一组具有相同类型的元素的集合，数组的长度及其类型都可作为等价类划分的依据。可划分为一个包含所有数组的有效等价类，一个空数组无效等价类，以及一个包含所有大于期望长度数组的无效等价类。

复合数据类型：复合数据类型是指包含两个或两个以上的相互独立的属性的输入数据。当对软件的一个组件模块(函数或对象)进行测试时，将使用这种输入了类型。对这种复合数据类型的输入进行等价类划分时，需要考虑输入数据的每个属性的合法与非法取值。

ii.关系与等价类划分

在集合论中，关系指的是一个 n 元组的集合。

如果规定了输入值的集合，或者规定了“必须如何”的条件，则可以定义一个有效等价类和一个无效等价类。

如果规定了输入数据的一组值，而且程序对不同输入值做不同处理，则可以定义若干有效等价类（每个值一个有效等价类）和一个无效等价类。

如果规定了输入数据必须遵守的规则，则可以定义一个有效等价类（符合规则）和若干无效等价类（从不同角度违反规则）。

iii.一元化分与多元化分

一元化等价类划分：每次只考虑一个输入变量，这样，每个输入变量形成了对输入域的一个划分，称为一元等价类划分，简称一元化分。程序有多少个变量，就有多少种划分，每个划分包含两个或两个以上的等价类。

多元划分：将所有输入变量的笛卡儿积作为程序的输入域，称为多元等价类划分，简称多元化分。此方法只产生一个划分，划分包含若干个等价类。

测试用例的选择常使用一元化分，因为一元化分较为简单且可量测。而多元化分所产生的等价类数量较大，并且其中有许多是无用的。

③ **划分等价类的步骤**

确定输入域：分析需求并确定所有的输入、输出量，以及变量类型和变量使用条件。

等价类划分：将每个变量的取值集合划分为互不相交的子集，每个子集对应一个等价类，所有的等价类就构成了对输入域的一个划分。

组合等价类：使用多元化方法，可以将等价类组合起来。

确定不可测的等价类：有些输入数据组合在实际测试过程中是无法生成的，包含这种数据的等价类就是不可测试等价类。不可测试数据指无法输入到被测软件中的那些数据组合。

④ **等价类的测试步骤**

i.划分等价类，形成等价类表；

ii. 为每个等价类规定一个唯一的编号；
iii. 设计一个新的测试用例，使其尽量多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止。
vi. 设计一个新的测试用例。使其覆盖一个而且只覆盖一个无效等价类，重复这一步，直到所有无效等价类均被覆盖为止。

等价类划分案例分析 1

某城市电话号码由三部分组成，分别是：

地区码—— 空白或三位数字；

前缀 —— 非‘0’或非‘1’开头的三位数字；

后缀 —— 4 位数字。

假定被测程序能接受一切符合上述规定的电话号码，拒绝所有不符合规定的电话号码。

(1) 划分等价类、列出等价类表

输入条件	有效等价类	编号	无效等价类	编号
地区码	空白	1	有非数字字符	5
			少于3位数字	6
	3位数字	2	多于3位数字	7
前缀	200~999	3	有非数字字符	8
			起始为’0’的3位数字	9
			起始为’1’的3位数字	10
			少于3位数字	11
			多于3位数字	12
后缀	4位数字	4	有非数字字符	13
			少于4位数字	14
			多于4位数字	15

一定要注意非数字字符的情况

(2)

测试用例编号	输入数据			预期输出	覆盖等价类
	地区码	前缀	后缀		
1	空白	321	4567	接受（有效）	1, 3, 4
2	123	805	9876	接受（有效）	2, 3, 4
3	20A	321	4567	拒绝（无效）	5
4	33	234	5678	拒绝（无效）	6
5	1234	234	4567	拒绝（无效）	7
6	123	2B3	1234	拒绝（无效）	8
7	123	013	1234	拒绝（无效）	9
8	123	123	1234	拒绝（无效）	10
9	123	23	1234	拒绝（无效）	11
10	123	2345	1234	拒绝（无效）	12
11	123	234	1B34	拒绝（无效）	13
12	123	234	34	拒绝（无效）	14
13	123	234	23345	拒绝（无效）	15

等价类划分案例分析 2

三角形问题的等价类测试。

输入三个整数 a、b 和 c 分别作为三角形的 3 条边，通过程序判断由这 3 条边构成的三条边类型是：等边三角形、等腰三角形、一般三角形或非三角形（不能够成一个三角形）。

假定 3 个输入 a、b 和 c 在 1～100 之间取值，三角形问题可以更详细地描述为：输入 3 个**整数** a、b 和 c 分别作为三角形的三条边，要求 a、b 和 c 必须满足以下条件。

c1：1≤a≤100
c2：1≤b≤100
c3：1≤c≤100
c4：a<b+c
c5：b<a+c
c6：c<a+b

输出下列 4 种情况之一：

①如果不满足条件 c4、c5 和 c6 中的一个，则程序输出为“非三角形”。

② 如果 3 条边相等，则程序输出为“等边三角形”。

③ 如果恰好有两条边相等，则程序输出为“等腰三角形”。

⑤ 如果 3 条边都不相等，则程序输出为“一般三角形”。

显然，这 4 种情况相互排斥。

从输入域进行分类，可以得到下图的三角形输入域等价类表

编号	输入条件（a,b,c）	有效等价类	无效等价类
E1	1≤a, b, c≤100的整数	是	
E2	1≤a, b, c≤100，且其中一边为小数		是
E3	1≤a, b, c≤100，且其中二边为小数		是
E4	1≤a, b, c≤100，且其中三边为小数		是
E5	其中一边小于1		是
E6	其中二边小于1		是
E7	其中三边小于1		是
E8	其中一边大于100		是
E9	其中二边大于100		是
E10	其中三边大于100		是
E11	只输入一个数		是
E12	只输入二个数		是
E13	输入三个以上的数		是
E14	输入非数值型的数据		是

测试用例编号	输入数			期望输出	对应等价类
	a	b	c		
Test1	5	6	7	一般三角形	E1
Test2	2.5	6	7	请输入1-100的三个整数	E2
Test3	2.5	3.5	7	请输入1-100的三个整数	E3
Test4	2.5	3.5	4.5	请输入1-100的三个整数	E4
Test5	0	6	7	请输入1-100的三个整数	E5
Test6	0	-1	7	请输入1-100的三个整数	E6
Test7	0	-1	0	请输入1-100的三个整数	E7
Test8	101	6	7	请输入1-100的三个整数	E8
Test9	101	102	7	请输入1-100的三个整数	E9
Test10	101	102	103	请输入1-100的三个整数	E10
Test11	5			请输入1-100的三个整数	E11
Test12	5	6		请输入1-100的三个整数	E12
Test13	5	6	7，8	请输入1-100的三个整数	E13
Test14	#	m	）	请输入1-100的三个整数	E14

若从输出域来对等价类进行划分，等价类划分表如下：

编号	输入条件（a,b,c）	有效等价类	无效等价类
E15	1≤a,b,c≤100，且三个相同的整数	是	
E16	1≤a,b,c≤100，且二个相同的整数	是	
E17	1≤a,b,c≤100，且三个不相同的整数	是	
E18	任一边数大于其它二边数之和的整数	是	

测试用例编号	输入数			期望输出	对应等价类
	a	b	c		
Test15	6	6	6	等边三角形	E15
Test16	6	6	5	等腰三角形	E16
Test17	3	4	5	一般三角形	E17
Test18	4	1	2	非三角形	E18

问题：在等价类划分时，必须要对输入域和输出域分别进行等价类划分吗

（2） **边值分析方法**

① 边界值分析方法概述

边界值分析法主要从数据的定义域的边界数据进行分析，对于合法与不合法的边界数据进行选取和测试。用来检查用户输入的信息、返回的结果以及中间计算结果是否正确。

② 边界值的获取及测试用例的设计

测试时输入变量取值：最小值(min)、略高于最小值(min+)、正常值(nom)、略低于最大值(max-)、最大值(max)

对于一个含有 n 个变量的程序，保留其中一个变量，让其余的变量取

正常值，被保留的变量依次取最小值(min)、略高于最小值(min+)、正常值(nom)、略低于最大值(max-)、最大值(max)，对每一个变量都重复进行。因此，对于一个有 n 个变量的程序，边界值分析测试程序就有 4n+1 个测试用例

测试所包含的边界值常见类型有：数值、字符、位置、大小、尺寸、空间等。

边界值的获取及生成测试用例的步骤：

（1）使用一元划分方法划分输入域。此时，有多少个输入变量就形成多少种划分

（2）为每种划分确定边界，也可利用输入变量之间的特定关系确定边界

（3）设计测试用例，确保每个边界至少出现在一个测试输入数据中

③ 健壮性测试

健壮性测试是边界分析测试的一种扩展，除了取上面已述的五种边界值外，还要考虑超出范围的值，即比最小值要小（min-）、比最大值要大（max+）的取值。对于一个含有 n 个变量的程序而言，同样，保留一个变量，让其余变量取正常值，这个保留的变量依次取七个值（min-、min、min+、nom、max-、max、max+），每个变量重复进行，则健壮性测试的用例将产生 6n+1 个测试用例。

边界分析法案例分析

考虑函数 findprice,它有两个整型输入变量，分别为 code 和 qty，其中 code 表示商品的编码，qty 表示采购数量。当函数 findprice 访问数据库，查询并显示 code 编码所对应的产品的单价、描述信息以及总的采购价格。当 code 和 qty 中任意一个为非法输入时，函数 findprice 显示一条错误提示信息并返回。假设编码 code 的有效区间为[99，999]，数量 qty 的有效输入区间为[1，100] 。

首先，为两个输入变量创建等价类。由上假设的区间可知有如下等价类，见下表：

变量	变量取值	等价类编号	备注
code	小于99	E1	无效等价类
	[99,999]	E2	有效等价类
	大于999	E3	无效等价类
qty	小于1	E4	无效等价类
	[1,100]	E5	有效等价类
	大于100	E6	无效等价类

根据相关边界值来设计测试用例（9+4），见下表：

测试用例编号	变量code	变量qty	预期输出
Test1	200	0	错误提示信息
Test2	200	1	相关单价等信息
Test3	200	2	相关单价等信息
Test4	200	50	相关单价等信息
Test5	200	99	相关单价等信息
Test6	200	100	相关单价等信息
Test7	200	101	错误提示信息
Test8	98	50	错误提示信息
Test9	99	50	相关单价等信息
Test10	100	50	相关单价等信息
Test11	998	50	相关单价等信息
Test12	999	50	相关单价等信息
Test13	1000	50	错误提示信息

【注：对实际问题要看是否增添健壮性测试】

（3） 决策表法

① 决策表法概述

决策表又称为判定表，是分析和表达多逻辑条件下执行不同操作的情况的工具。能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏，设计出完整的测试用例集合。在所有功能性测试方法中，基于决策表的测试方法是最严格的测试方法之一。

② 决策表的组成

决策表的组成：由条件桩、动作桩、条件项、动作项四个部分组成，如图所示：

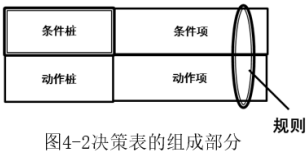


图4-2决策表的组成部分

条件桩：列出了问题的所有条件，通常认为列出的条件的次序无关紧要。
动作桩：列出了问题规定可能采取的操作。这些操作的排列顺序没有约束
条件项：列出了针对它左列条件的取值。在所有可能情况下的真假值
动作项：列出在条件项的各种取值情况下应该采取的动作
规 则 ：任何一个条件组合的特定取值及其相应要执行的操作称为规则

③ 决策表的类型

有限条目决策表：所有条件都是二叉条件（真/假）
扩展条目决策表：条件可以有多个值

④ 决策表的建立步骤

i. 列出所有的条件桩和动作桩
ii. 确定规则的个数
iii. 填入条件项
vi. 填入动作项，得到初始决策表
v. 合并相似规则，得到优化决策表

⑤ 决策表的简化

简化是以合并相似规则为目标；若表中有两条以上的规则具有相同的动作，并且在条件项之间存在记为相似的关系，便可以合并

⑥ 决策表的优点

能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏，因此，利用决策表能够设计出完整的测试用例集合。最为严格、最具逻辑性的测试方法

决策表法案例分析

对三角形问题，使用决策表法来设计其测试用例。

i. 列出条件桩和动作桩

ii. 确定规则的个数

条件桩	动作桩	规则个数
C1: 1≤a≤100	非三角形	
C2: 1≤b≤100	不等边三角形	
C3: 1≤c≤100	等腰三角形	
C4: a=b?	等边三角形	
C5: b=c?	不可能	$2^6=64$
C6: a=c?		

iii. 填入条件项，见下表：F 表示取假，T 表示取真

vi. 填入动作项

C1：1≤a≤100	F	F	F	F	F	F	F	F	F	F	……
C2：1≤b≤100	F	F	F	F	F	F	F	F	F	F	……
C3：1≤c≤100	F	F	F	F	F	F	F	F	T	T	……
C4：a=b?	F	F	F	F	T	T	T	T	F	F	……
C5：b=c?	F	F	T	T	F	F	T	T	F	F	……
C6：a=c?	F	T	F	T	F	T	F	T	F	T	……
非三角形	√	√	√	√	√	√	√	√	√	√	……
不等边三角形											……
等腰三角形											……
等边三角形											……
不可能											……

v. 合并相似规则后，如下表：

C1：1≤a≤100	F	T	T	T	T	T	T	T	T	T	T
C2：1≤b≤100	–	F	T	T	T	T	T	T	T	T	T
C3：1≤c≤100	–	–	F	T	T	T	T	T	T	T	T
C4：a=b?	–	–	–	T	T	T	T	F	F	F	F
C5：b=c?	–	–	–	T	T	F	F	T	T	F	F
C6：a=c?	–	–	–	T	F	T	F	T	F	T	F
非三角形	√	√	√								
不等边三角形											√
等腰三角形							√		√	√	
等边三角形				√							
不可能					√	√		√			

iv. 根据决策表设计测试用例

测试用例编号	a	b	c	预期输出
Test1	4	1	2	非三角形
Test2	1	4	2	非三角形
Test3	1	2	4	非三角形
Test4	6	6	6	等边三角形
Test5	?	?	?	不可能
Test6	?	?	?	不可能
Test7	6	6	7	等腰三角形
Test8	?	?	?	不可能
Test9	6	7	6	等腰三角形
Test10	7	6	6	等腰三角形
Test11	3	4	5	不等边三角形

（4） **因果图法**

① 因果图方法概述

因果图，也称作依赖关系模型。主要用于描述软件输入条件（原因）与软件输出结果（结果）之间的依赖关系。“原因”是指软件需求中能影响软件输出的任意输入条件。“结果”是指软件对某些输入条件的组合所做出的响应。可以是一条提示信息，也可以是弹出的一个新窗口，还可以是数据库的一次更新。结果可以可见或不可见。

因果图法特别适用于被测程序具有多种输入条件，程序的输出又依赖于输入条件的各种组合的情况。因果图方法最终生成的就是判定表。

② 因果图中的基本符号和约束

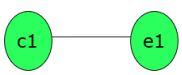
在图果图有两种类型的符号：

(1)因果关系符号有：对应关系、否定关系、选择关系和并列关系；

(2)约束关系符号有：互斥关系、包含关系、唯一关系、要求关系和屏蔽关系；

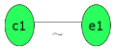
类别	名 称	图 符	含 义
因果关系	一一对应关系	原因 ○——○ 结果	原因出现，则结果出现；反之亦然。
	否定关系	原因 ○~~○ 结果	原因出现，结果不出现；原因不出现，结果出现。
	选择关系	原因1、原因2 ○ 结果	若几个原因中有一个出现，则结果出现；只有当几个原因都不出现时，结果才不出现。
	并列关系	原因1、原因2 ○ 结果	若几个原因同时出现，则结果出现；若几个原因有一个不出现，则结果不出现。
约束关系	E关系（输入）（互斥关系）	E ○ 原因a、原因b	表示 a、b 两个原因不会同时成立，两个中最多有一个可能成立。
	I关系（输入）（包含关系）	I ○ 原因a、原因b、原因c	表示 a、b 和 c 三个原因中至少有一个必须成立。
	O关系（输入）（唯一关系）	O ○ 原因a、原因b	表示 a 和 b 中必须有一个且仅有一个成立。
	R关系（输入）（要求关系）	R ○ 原因a、原因b	表示 a 出现时 b 必须出现。
	M关系（输出）（屏蔽关系）	结果a、结果b ○ M	表示 a 出现时 b 不能出现。

恒等：若 C1 是 1，则 e1 也是 1，否则 e1 为 0，

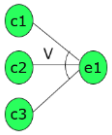


，

举例：如果密码正确，则登陆系统。

非：若 c1 是 1，则 e1 为 0，否则 e1 为 1，用符号~表示。 

举例：如果密码不正确，则给出提示信息

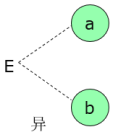
或：若 c1 或 c2 或 c3 是 1，则 e1 是 1，否则 e1 为 0，“或”可有任意个输入，用符号“V”表示。 

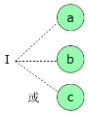
举例:如果是系统管理员或者是班主任，可以直接查看成绩

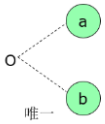
与：若 c1 和 c2 都是 1，则 e1 为 1，否则 e1 为 0，“与”也可有任意个输入。用符号“∧”

举例：如果用户名正确，并且密码正确，才能登陆系统。

E 约束(异)：a 和 b 中最多有一个可能为 1，即 a 和 b 不能同时为 1

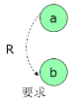


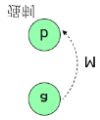
表示：条件 a,b 不能同时为 1




I 约束(或)：a、b、c 中至少有一个必须是 1，即 a、b、c 不能同时为 0

O 约束(唯一)：a 和 b 必须有一个且仅有一个为 1

R 约束(要求)：a 是 1 时，b 必须是 1 



M 约束(强制)：若结果 a 是 1，则结果 b 强制为 0

③ 因果图测试用例的设计步骤

i. 分析程序规格说明中哪些是原因，哪些是结果。原因常常是输入条件或输入条件的等价类，结果则是输出条件。

ii. 分析程序规格说明中描述内容的语义和限制，找出两类关系，画出因果图。

iii. 把因果图转换成判定表。

iv. 对判定表的每一列写成一个测试用例。

④ 使用因果图法的优点

i. 考虑到了输入情况的各种组合以及各个输入情况之间的相互制约关系。

ii. 能够帮助测试人员按照一定的步骤，高效率的开发测试用例。

iii. 因果图法是将自然语言规格说明转化成形式语言规格说明的一种严格的方法，可以指出规格说明存在的不完整性和二义性。

因果图法案例分析

假设某软件中对一些文件名要求如下：

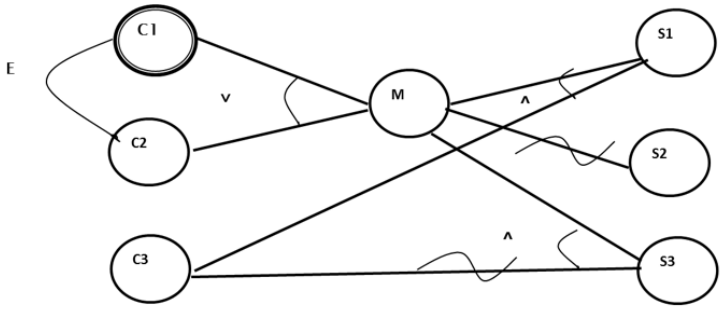
对于输入或输出文件名规定：第一个字符必须是字母 I 或 O（如：I 字符开始表示输入文件名，O 字符开始表示输出文件名），第二个字符必须是一个数字，如：I1、O3 等。当输入文件名后，对文件进行相关处理。如果文件名中第一个字符不正确，则给出“操作文件类型错”信息。若第一个字符正确，但第二个字符不正确，则给出“文件顺序号错”信息。

i.根据规格需求，列出原因和结果

原因：C1:第一个字符是 I　C2:第一个字符是 O　C3:第二个字符是数字

结果：S1：对文件进行处理 S2：给出“操作类型错”的信息　S3：给出“文件顺序号错的信息”

ii.画出因果图，找出约束关系



iii.将因果图转换为判定表，其中 T 表示是，F 表示否

条件	C1:第一个字符是I	T	F	F	F	T	F
	C2:第一个字符是O	F	T	F	F	F	T
	C3:第二个字符是数字	T	T	F	T	F	F
操作	S1:对文件进行处理	√	√				
	S2:操作类型错			√	√		
	S3:顺序号错					√	√

iv.根据判定表设计测试用例

测试用例编号	输入文件名	预期输出
Test1	I4	对文件进行处理
Test2	05	对文件进行处理
Test3	XX	操作类型错
Test4	X8	操作类型错
Test5	IX	顺序号错
Test6	0X	顺序号错

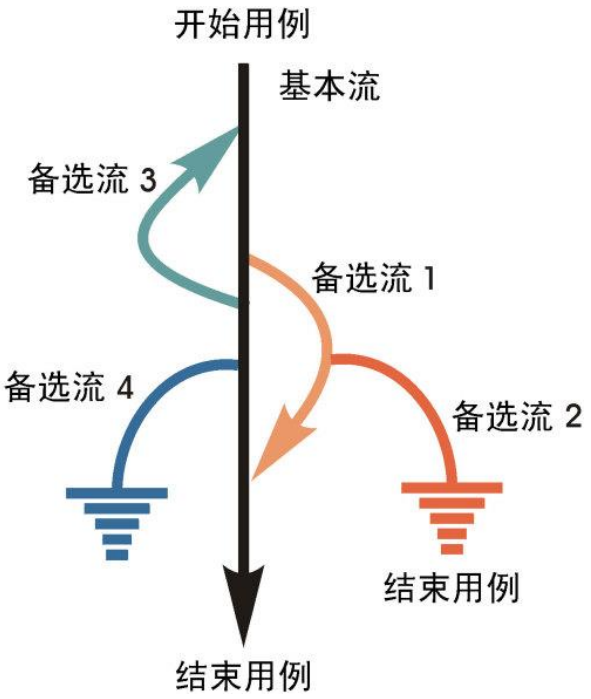
（5） **场景法**

现在的软件几乎都是用事件触发来控制流程的，比如 GUI 软件、游戏等，事件触发时的情景形成了场景，而同一事件不同的出发顺序和处理结果就形成了事件流。

① 基本概念

i.在测试一个软件的时候，在场景法中，测试流程是软件功能按照正确的时间流实现的一条正确流程，那么我们把这个称为该软件的基本流。

ii.出现故障或缺陷的过程，就用备选流加以标注，这样，备选流就可以是从基本流来的，或是由备选流中引出的



直黑线表示基本流，是经过用例的最简单的路径。备选流用不同的色彩表示，一个备选流可能从基本流开始，在某个特定条件下执行，然后重新加入基本流中；也可能起源于零一个备选流，或者终止用例而不再重新加入到某个流。

② 场景法设计测试用例的步骤

i. 根据说明，描述出程序的基本流及各项备选流

ii. 根据基本流和各项备选流生成不同的场景

iii. 对每个场景生成相应的测试用例

iv. 对生成的所有测试用例重新复审，去掉多余的测试用例，测试用例确定后面对每一个测试用例确定测试数据值。

场景法案例分析

在当当网网上书店都订购过书籍，整个订购过程为：用户登录到网站后，进行书籍的选择，当选好自己心仪的书籍后进行订购，这时把所需图书放进购物车，等进行结帐的时候，用户需要登录自己注册的帐号，登录成功后，进行结帐并生成订单，整个购物过程结束。

① 确定基本流和备选流

基本流：用户到网站，选择书籍，进行订购，把所需图书放入购物车，等进行结账的时候，登陆自己的账号，登陆成功后，生成订单

备选流 1：账号不存在

备选流 2：账号错误

备选流 3：密码错误

备选流 4：无选购书籍

备选流 x：退出系统

② 确定场景

场景 1——购物成功 ---------- 基本流

场景 2——账号不存在 --------- 基本流、备选流 1

场景 3——账号错误 ---------- 基本流、备选流 2

场景 4——密码错误 ---------- 基本流、备选流 3

场景 5——无选购书籍 --------- 基本流、备选流 4

③ 确定测试用例

对于每一个场景都需要确定测试用例。可以采用矩阵或决策表来确定和管理测试用例

对于每个测试用例，存在一个测试用例 ID、条件(或说明)、测试用例中设计的所有数据元素（作为输入或已经存在于数据库中）以及预期结果

ID	场景/条件	帐号	密码	选购书籍	预期结果
1	场景1：购物成功	V	V	V	成功购物
2	场景2：帐号不存在	I	n/a	n/a	提示帐号不存在
3	场景3：帐号错误	I	V	n/a	提示帐号错误，返回基本流步骤2
4	场景4：密码错误	V	I	n/a	提示密码错误，返回基本流步骤3
5	场景5：无选购书籍	V	V	I	提示选购书籍，返回基本流步骤5

ID	场景/条件	帐号	密码	选购书籍	预期结果
1	场景1：购物成功	xu	123456	《软件测试终极宝典》	成功购物
2	场景2：帐号不存在	zhang	n/a	n/a	提示帐号不存在
3	场景3：帐号错误	zhou	123456	n/a	提示帐号错误，返回基本流步骤2
4	场景4：密码错误	xu	123$%^	n/a	提示密码错误，返回基本流步骤3
5	场景5：无选购书籍	xu	123456	空	提示选购书籍，返回基本流步骤5

（6） **其他黑盒测试方法**

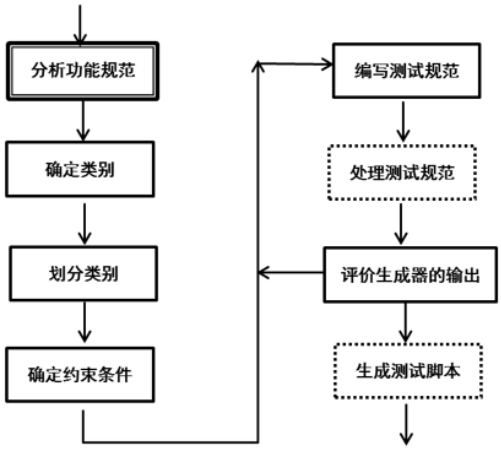
①类别划分法

i. 类别划分法概述

类别划分法是一种从软件需求生成测试用例的系统化的方法。该方法同时可以包含手工和自动完成的步骤。类别划分法的本质是测试人员将软件需求转换为相应的测试规范，其中，测试规范由对应于软件输入变量和环境对象的各种类别构成。

每个类别被划分为若干个对应于软件输入变量、环境对象状态的一个或多个取值的选项。测试规范中同时也包含了各选项之间的关系，以便确保生成合理、有效的测试集。将编写好的测试规范输入测试框架生成器，获得相应的框架，再根据测试框架可生成相应的测试脚本。

ii. 类别划分法的步骤



②谓词测试

i. 谓词测试概述

规则可以形式化地表示为谓词。例如，考虑软件需求“若打印机处于 ON 状态且具备打印纸，则发送要打印的文件”。这需求中包含一个条件和一个动作。而条件是一个关系表达式，即打印机在打印状态和有打印纸存在。可以表示为 P：

P：(printer_status=ON)∧(printer_tray= ¬empty)

这是一个谓词 P，它是由布尔运算符“∧”连接的关系表达式。编程人员可能正确地为这个谓词编码，也可能没有正确编码。

根据谓词产生测试用例，来测试程序中的错误，从而可以确保在测试

中发现某种类型的所有缺陷。这种用于验证谓词实现是否正确的测试称为谓词测试。

ii.谓词测试中的故障类型

一个条件可以表示成简单谓词或复合谓词。**简单谓词**就是一个布尔变量或关系表达式，其中变量可能取非。**复合谓词**可以是一简单谓词，或是由若干简单谓词或其补通过二元布尔运算符连接起来的式子。

谓词测试，主要关注三类故障：布尔运算符故障、关系运算符故障、算术表达式故障。

iii.谓词测试的准则

BOR:对于复合谓词 P，如果测试集 T 确保能够检测出 P 实现中存在的单/多布尔运算符故障，则 T 满足了 BOR 测试准则。称 T 为 BOR 充分测试集。

BRO:对于复合谓词 P，如果测试机 T 确保能够检测出 P 实现中存在的单/多布尔运算符及关系运算符故障，则 T 满足了 BRO 测试准则。称 T 为 BRO 充分测试集。

BRE: 对于复合谓词 P，如果测试机 T 确保能够检测出 P 实现中存在的单/多布尔运算符、关系运算符及算数表达式故障，则 T 满足了 BRE 测试准则。称 T 为 BRE 充分测试集。

③错误推测法

基本思想是：利用直觉和经验推测软件系统中可能出错的类型，列举出程序中所有可能的错误和容易发生错误的情况，用清单的形式表示，然后，再根据清单来编写测试用例。

常从以下几个方面来推测软件件系统中存在的错误：

(1)软件产品以前版本中已存在的未解决的问题；

(2)因为编程语言、操作系统、浏览器等环境的限制而出现的问题；

(3)因模块间关联的测试出现的缺陷，修复后可能带来其他的问题等。

4、黑盒测试的依据

黑盒测试也成为功能测试、行为测试或数据驱动测试，在测试时，把程序看作一个不能打开的黑盒，测试人员完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合他的功能说明。

黑盒测试主要检查以下几个方面的内容：

（1） 正确性：计算结果，命名方面

（2） 可用性：是否可以满足软件的需求说明

（3） 边界条件：输入部分的边界值

（4） 性能：程序的性能取决于两个因素：运行速度的快慢和需要消耗的系统资源

（5） 压力测试：多用户情况可以考虑使用压力测试工具，建议将压力和性能测试结合起来进行。如果有负载平衡的话还要在服务器端打开检验工具，查看服务器 CPU 使用率，内存占用情况，如果有必要可以模拟大量数据输入，对硬盘的影响等等信息。

（6） 错误恢复：错误处理、页面数据验证，包括突然间断点，输入错误数据等

（7） 安全性测试：对系统的安全进入、安全操作及相关权限进行测试。特别是一些商务网站，或者跟钱有关，或者和公司秘密有关的 web 更是需要这方面的测试

（8） 兼容性：不同浏览器，不同应用程序版本在实现功能时的表现

黑盒测试的流程

（1）测试计划 （2）测试设计 （3）测试开发 （4）测试执行（5）测试评估

5、黑盒测试综合策略

（1）首先用边界值分析法设计测试用例

（2）必要时用等价类划分法补充测试用例

（3）必要时再用猜错法补充测试用例

（4）如果在程序的说明中含有输入条件的组合，宜在一开始就采用因果图法，然后再按上述步骤进行

6、黑盒测试运用实例

保险金计算程序：

保险金=500 ×年龄系数 - 安全驾驶折扣

安全驾驶折扣是投保人驾驶执照上当前点数的函数,年龄系数是投保人年龄的函数，若点数低于等于与年龄有关的点数门限，则给予安全驾驶折扣。

程序输入:年龄、点数，驾驶人年龄范围为 16-100 岁；点数范围为 0-12。

输出：保险金。

年龄范围	年龄系数	门限点数	安全驾驶折扣
16≤年龄<25	2.8	1	50
25≤年龄< 35	1.8	3	50
35 ≤年龄< 45	1.0	5	100
45≤年龄< 60	0.8	7	150
60≤年龄≤ 100	1.5	8	200

解：

使用边界值测试法

变量	min	min+	nom	max	max+
年龄	16	17	50	99	100
点数	0	1	6	11	12

变量	min	min+	nom	max	max+
年龄	16	17	20	24	-
年龄	25	26	30	34	-
年龄	35	36	40	44	-
年龄	45	46	53	59	-
年龄	60	61	75	99	100
点数	0	-	-	-	1
点数	2	-	-	-	3
点数	4	-	-	-	5
点数	6	-	-	-	7
点数	8	9	10	11	12

等价类划分法测试

年龄等价类集合

A1:{16 ≤ 年龄 ≤ 25}

A2:{25 ≤ 年龄 < 35}

A3:{35 ≤ 年龄 < 45}

A4:{45 ≤ 年龄 < 60}

A5:{60 ≤ 年龄 ≤ 100}

点数等价类集合

P1:{点数 = 0，1}

P2:{点数 = 2，3}

P3:{点数 = 4，5}

P4:{点数 = 6，7}

P5:{点数 = 8，9，10，11，12}

测试用例编号	年龄	点数	测试用例编号	年龄	点数
Test1	18	0	Test14	40	6
Test2	18	2	Test15	40	10
Test3	18	4	Test16	50	0
Test4	18	6	Test17	50	2
Test5	18	10	Test18	50	4
Test6	30	0	Test19	50	6
Test7	30	2	Test20	50	10
Test8	30	4	Test21	80	0
Test9	30	6	Test22	80	2
Test10	30	10	Test23	80	4
Test11	40	0	Test24	80	6
Test12	40	2	Test25	80	10
Test13	40	4			

决策表法测试

年龄	16–25	16–25	25–35	25–35	35–45	35–45	45–60	45–60	60–100	60–100
点数	0–1	2–12	0–3	4–12	0–5	6–12	0–7	8–12	0–5	6–12
年龄系数	2.8	2.8	1.8	1.8	1	1	0.8	0.8	1.5	1.5
安全驾驶折扣	50	0	50	0	100	0	150	0	200	0

测试用例编号	年龄	点数	测试用例编号	年龄	点数
Test1	18	0	Test6	40	6
Test2	18	2	Test7	50	7
Test3	30	0	Test8	50	8
Test4	30	4	Test9	80	5
Test5	40	5	Test10	80	6