第5章

黑盒测试

本章内容

- 黑盒测试方法的基本概念
- 等价类划分
- 边界值分析
- 因果图
- 判定表

学习目标

(1) 理解黑盒测试方法的基本概念。

(2) 了解黑盒测试的常用策略。

(3) 掌握黑盒测试中等价类划分、边界值分析、因果图、判定表驱动测试技术。

(4) 理解黑盒测试中等价类划分和边界值分析、因果图和判定表之间的区别和联系。

(5) 根据规格说明运用等价类划分和边界值分析方法进行测试用例设计。

(6) 根据规格说明运用因果图和判定表方法进行测试用例设计。

5.1 黑盒测试基本概念

黑盒测试是一种从软件外部对软件实施的测试，也称功能测试、数据驱动测试或基于规格说明的测试。其基本观点是：任何程序都可以看作是从输入定义域到输出值域的映射，将被测程序看作一个打不开的黑盒，黑盒里面的内容(实现)是完全不知道的，也不关心黑盒里面的结构，只知道软件要做什么，只关心软件的输入数据和输出结果。

黑盒测试用来检测每个功能是否都能正常使用。在测试中，在完全不考虑程序内部结构和内部特性的情况下，在程序接口进行测试，它只检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据而产生正确的输出信息。因此，黑盒测试着眼于程序外部结构，主要针对软件界面和软件功能进行测试。如果外部特性本身设计有问题或规格说明的规定有误，用黑盒测试方法是发现不了的。

黑盒测试是以用户的角度，从输入数据与输出数据的对应关系出发进行测试的，能更好、更真实地从用户角度来考察被测系统的功能性需求实现情况。黑盒

测试方法在软件测试的各个阶段，如单元测试、集成测试、系统测试及验收测试等阶段中都发挥着重要作用，尤其在系统测试和确认测试中，其作用是其他测试方法无法取代的。

锲而不舍，金石可镂

黑盒测试是针对产品说明开展的外部检测，要求测试员对待问题要锲而不舍，并善于总结经验。

黑盒测试方法把产品软件想象成一个只有出口和入口的黑盒，在测试过程中，只需要知道向黑盒输入什么，黑盒会产生什么结果。因此，黑盒测试方法是在程序接口上进行的测试，主要是为了发现以下错误。

（1）是否有功能错误，是否有功能遗漏。

（2）是否能够正确地接收输入数据并产生正确的输出结果。

（3）是否有数据结构错误或外部信息访问错误。

（4）是否有程序初始化和终止方面的错误。

黑盒测试方法主要有等价类划分法、边界值分析法、错误推测法、因果图法、判定表驱动法、正交实验设计法、功能图法、场景法等。

5.2　等价类划分

“黑盒”方法是详尽的输入测试，只有当所有可能的输入都用作测试条件时，才能以这种方式检测程序中的所有错误。理想的测试，是从所有可能的输入中找出代表性输入数据，力求使用最少的测试数据，发现尽可能多的缺陷，达到最好的测试质量。

确定这样的输入子集需要借助测试用例的两个特性：测试用例数量达到最少；某个测试用例要能覆盖大部分其他测试用例。第二个特性就暗示我们，应该尽量将程序输入范围进行划分，将其划分为有限数量的等价类，这样就可以合理地假设测试每个等价类的代表性数据等同于测试该类的其他任何数据。这两种特性形成了称为等价类划分的黑盒测试方法。

等价类是指某个输入测试的子集合，等价类划分就是根据需求规则把输入域划分为不同的子集合，如图 5-1 所示。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的，并合理地假定：测试某等价类的代表值就等于对这一类其他值的测试。因此，可以把全部输入数据合理划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件，就可以用少量代表性的测试数据，取得较好的测试结果。

黑盒测试方法不仅要测试所有合法的输入，还要测试那些非法但可能的输入。因此，等价类划分可有两种不同的情况：有效等价类和无效等价类。

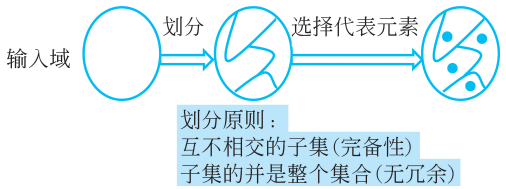

图 5-1 等价类划分

(1) 有效等价类：有效等价类就是有效值的集合，它们是符合程序要求、合理且有意义的输入数据。

(2) 无效等价类：无效等价类就是无效值的集合，它们是不符合程序要求、不合理或无意义的输入数据。

例如，某需求规格说明书对某功能的规定：红酒单价 500 元一瓶，每次最多购买 100 瓶，购买数量在 50 瓶以上时 450 元一瓶。

此时，决定红酒价格的就是购买数量，50 瓶以内还是 50 瓶以上。因此，在限制输入数据是正整数的前提下，可以划分(0,50)、[50,100]两个有效等价类，以及(100,—)一个无效等价类。黑盒测试时，可以分别从(0,50)、[50,100]两个数据区间选择代表性数据作为有效输入数据，例如 25,75；从(100,—) 数据区间选择代表性数据作为无效输入数据，例如 150。

从上面的简单示例中可以看出，等价类划分法主张从大量的数据中选择一部分数据用于测试，即尽可能使用最少的测试用例覆盖最多的数据，以发现更多的软件缺陷。该方法背后的假设是，如果某个等价类中的一个条件/值通过，则所有其他条件/值也将通过。同样，如果某个等价类中的一个条件失败，则该等价类中的所有其他条件都将失败。

了解了有效等价类与无效等价类，那么如何划分等价类呢？

5.2.1 等价类划分原则

在给定了输入或外部条件之后，等价类的划分原则如下。

(1) 按照区间划分：如果程序要求输入值是一个有限区间的值，则可以将输入数据划分为一个有效等价类和两个无效等价类，有效等价类为指定的取值区间，两个无效等价类分为有限区间两边的值。

例如，某程序要求输入值 x 的范围为[1,100]，则有效等价类为 $1 \leqslant x \leqslant 100$，无效等价类为 $x<1$ 和 $x>100$。

(2) 按照数值集合划分：如果程序要求输入的值是一个“必须成立”的情况，则可以将输入数据划分为一个有效等价类和一个无效等价类。例如，某程序要求密码正确，则正确的密码为有效等价类，错误的密码为无效等价类。

例如，某程序要求输入数据 x 必须属于固定的枚举类型{1,9,15}，且对这三个值做统一处理，则可以划分一个有效等价类 $x=1,9,15$ 和一个一个无效等价类 $x \neq 1,9,15$ 的输入值集合。

(3) 按照数值划分：如果程序要求输入的值是一组数据，并且程序要对每一个输入值分别进行处理的情况，则可以将输入数据划分为 n 个有效等价类(每个输入值确定一个有效等价类)和一个无效等价类(所有不允许的输入值的集合)。

例如，某程序要求输入数据 x 必须属于固定的枚举类型{1,9,15}，且对这三个值分别

进行了处理，则可以划分三个有效等价类 $x=1$，$x=9$，$x=15$ 和一个 1 个无效等价类 $x\neq1$，9，15 的输入值集合。

（4）按照规则划分：如果程序要求输入数据是一组可能的值，或者要求输入值必须符合某个条件，则可以将输入的数据划分为一个有效等价类和一个无效等价类。

例如，某程序要求输入数据必须是以数字开头的字符串，则以数字开头的字符串是有效等价类，不是以数字开头的字符串是无效等价类。

（5）细分等价类：如果在某一个等价类中，每个输入数据在程序中的处理方式都不相同，则应将该等价类划分成更小的等价类，并建立等价表。

同一个等价类中的数据发现程序缺陷的能力是相同的，如果使用等价类中的一个数据不能捕获缺陷，那么使用等价类中的其他数据也不能捕获缺陷；同样，如果等价类中的一个数据能捕获缺陷，那么该等价类中的其他数据也能捕获缺陷，即等价类中的所有输入数据都是等效的。

5.2.2　多变量的等价类划分组合

正确地划分等价类可以极大地降低测试用例的数量，测试会更准确有效。划分等价类时，要认真分析、审查划分，过于粗略的划分可能会漏掉软件缺陷。如果错误地将两个不同的等价类当作一个等价类，则会遗漏测试情况。例如，某程序要求输入取值范围为 1～100 的整数，若一个测试用例输入了数据 0.6，则在测试中很可能只检测出非整数错误，而检测不出取值范围的错误。

在输入域包含多个变量时，首先需要针对每个变量划分等价类，然后根据不同变量等价类的组合设计测试用例。此时，根据测试用例的完整性可以划分为弱等价类测试和强等价类测试；根据是否考虑到无效等价类的情况划分为一般等价类和健壮等价类。组合起来，测试用例的完整性可以分为以下四种情况。

（1）弱一般等价类测试：遵循单缺陷原则，要求用例覆盖每一个变量的一种取值即可，取值为有效值。如图 5-2 所示，图中假设函数 $y=f(x_1,x_2)$ 输入变量的取值范围分别为：$x_1\in[a,b]$ 或 $x_1\in[b,c]$ 或 $x_1\in[c,d]$，$x_2\in[g,f]$ 或 $x_2\in[f,e]$。

（2）弱健壮等价类测试：在弱一般等价类的基础上，增加取值为无效值的情况。为每个变量的每个无效等价类设计一个无效测试用例，其余变量的取值保持在有效范围内，如图 5-3 所示，其中，实心小圆圈代表有效等价类用例，空心小圆圈代表无效等价类用例。

（3）强一般等价类测试：遵循多缺陷原则，要求测试用例覆盖每个变量的每种有效取值之间的笛卡儿积，即所有变量所有有效取值的所有组合，如图 5-4 所示，变量 x_1 有 3 个有效等价类，变量 x_2 有 2 个有效等价类，那么应设计 3×2 共 6 个用例覆盖所有的有效等价类组合。

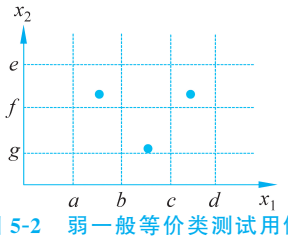

图 5-2　弱一般等价类测试用例

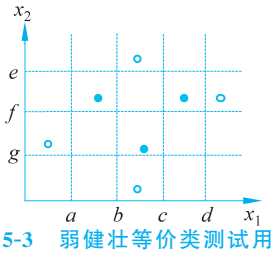

图 5-3　弱健壮等价类测试用例

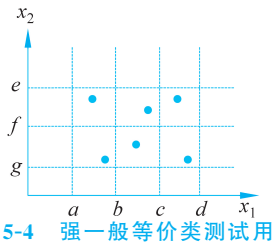

图 5-4　强一般等价类测试用例

(4) 强健壮等价类测试：在强一般等价类的基础上，增加取值为无效值的情况。要求测试用例覆盖每个变量的每种等价(有效和无效)取值之间的笛卡儿积，即所有变量所有等价(有效和无效)取值的所有组合，如图 5-5 所示，变量 x_1 有 3 个有效等价类和 2 个无效等价类，变量 x_2 有 2 个有效等价类和 2 个无效等价类，那么应设计(3+2)×(2+2)共 20 个用例覆盖所有的等价类组合。

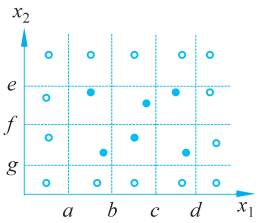

图 5-5 强健壮等价类测试用例

5.2.3 等价类划分测试

用等价类划分方法进行黑盒测试用例设计时，首先应该根据需求说明进行等价类划分；其次为有效等价类设计测试用例；最后为无效等价类设计测试用例。

在设计等价类设计测试用例时，通常遵循以下原则。

(1) 为每一个等价类规定一个唯一的编号。

(2) 设计一个新的测试用例，使其**尽可能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。

(3) 设计一个新的测试用例，使其**仅覆盖一个尚未被覆盖的无效等价类**，重复这一步，直到所有的无效等价类都被覆盖为止。

例 5-1 电话号码问题的等价类划分测试。

某城市电话号码由三部分组成：①地区码，空白或三位数字；②前缀，不能是以 1 和 2 开头的三位数字；③后缀，4 位数字。

解答：

(1) 划分等价类。此例中，可以把输入数据看作由地区码、前缀和后缀 3 个变量组成。根据需求说明划分为如表 5-1 所示的等价类。

表 5-1 某城市电话号码的等价类划分

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

（2）为有效等价类和无效等价类编写测试用例。

根据多变量等价类划分组合，为有效等价类编写测试用例时有以下 4 种选择。

① 弱一般等价类用例。

使用最少测试用例覆盖每个有效等价类。在例 5-1 中弱一般等价类的用例设计如表 5-2 所示。

表 5-2　弱一般等价类用例设计

用例编号	输　入			预 期 输 出	覆盖有效等价类
	地区码	前　缀	后　缀		
1	空白	300	1111	有效	(1)(3)(4)
2	232	555	2222	有效	(2)(3)(4)

② 弱健壮等价类用例。

在弱一般等价类的基础上，增加取值为无效值的情况。对于无效输入，测试用例将拥有一个无效值，并保持其余的值是有效的。在例 5-1 中弱健壮等价类的用例设计如表 5-3 所示。

表 5-3　弱健壮等价类用例设计

用例编号	输　入			预 期 输 出	覆盖有效等价类
	地区码	前　缀	后　缀		
1	空白	300	1111	有效	(**1**)(**3**)(**4**)
2	232	555	2222	有效	(**2**)(3)(4)
3	321	123	4006	无效	(**5**)(3)(4)
4	11	122	6330	无效	(**6**)(3)(4)
5	1235	133	6550	无效	(**7**)(3)(4)
6	空白	331	1234	无效	(1)(**8**)(4)
7	空白	033	3344	无效	(1)(**9**)(4)
8	333	144	3232	无效	(2)(**10**)(4)
9	444	14	3443	无效	(2)(**11**)(4)
10	555	1444	5201	无效	(2)(**12**)(4)
11	空白	255	4651	无效	(1)(3)(**13**)
12	321	456	652	无效	(2)(3)(**14**)
13	354	965	6422	无效	(2)(3)(**15**)

③ 强一般等价类用例。

强一般等价类是基于多缺陷假设，强一般等价类的测试用例是要覆盖每个有效等价类取值的笛卡儿积，即在有效等价类取值的所有组合。本例中强一般等价类测试用例的组合个数是 $2\times1\times1=2$，与弱一般等价类测试用例没有区别。

④ 强健壮等价类用例。

在强一般等价类的基础上,增加取值为无效值的情况,也是运用笛卡儿积思路得出测试用例。测试用例个数有(2+3)×(1+5)×(1+3)=5×6×4=120(个),用例数量太多,就不一一列出来了。

例 5-1 从四个不同方面来思考怎样设计等价类测试用例,实际工作中应该根据测试项目需求选择其中一种等价类测试用例进行测试。例如,如果测试的是核心模块,可以选择强健壮等价类测试用例进行测试;如果测试的是重要非核心模块,可以选择强一般等价类测试用例进行测试。本书中如果没有特殊说明,默认只进行强一般等价类测试用例设计。

知识拓展

使用等价类划分法设计测试用例的重点在于划分有效等价类和无效等价类粗细的粒度。粒度越粗,设计测试用例越少,粒度越细,设计测试用例越多。相对来说,粒度越细,越能发现更多问题。

例 5-2-1 NextDate 函数的等价类划分测试。

NextDate 函数包含三个变量:month、day 和 year,函数的输出为输入日期后一天的日期。例如,输入为 2020 年 3 月 7 日,则函数的输出为 2020 年 3 月 8 日。要求输出变量 month、day 和 year 均为整数值,并且满足下列条件:1≤month≤12,1≤day≤31,1920≤year≤2050。

解答:

该函数的主要特点是输入变量之间的逻辑关系比较复杂,变量 year 和变量 month 取不同值时,对应的变量 day 会有不同的取值范围,或 1～30 或 1～31 或 1～28 或 1～29。

(1) 划分等价类。

等价关系的要点是:等价类中的元素要被“同样处理”,此例中月中、月末、年末被“同样处理”。因此,更详细的有效等价类如下。

变量 month:M1={month:month 有 30 天}、M2={month:month 有 31 天,除去 12 月}、M3={month:month 是 2 月}、M4={month:month 是 12 月}。

变量 day:D1={day:1≤day≤28}、D2={day:day=29}、D3={day:day=30}、D4={day:day=31}。

变量 day:Y1={year:year 是闰年}、Y2={year:year 是平年}。

(2) 为有效等价类和无效等价类编写测试用例。

本例中只进行强一般等价类测试用例设计。强一般等价类要做独立性假设,以等价类的笛卡儿积表示。在本问题中,变量 month 等价类数量为 4、变量 day 等价类数量为 4、变量 year 等价类数量为 2,故强一般等价类测试用例数量为 4×4×2=32,如表 5-4 所示。

从表 5-4 中可以看到,NextDate 函数对输入日期的输出处理主要有以下三种情况。

表 5-4 NextDate 函数的强一般等价类用例设计

用例编号	输入			预期输出	覆盖有效等价类
	year	month	day		
1	1999	4	15	1999-4-16	Y1 M1 D1
2	1999	4	29	1999-4-30	Y1 M1 D2

续表

用例编号	输　入			预 期 输 出	覆盖有效等价类
	year	month	day		
3	1999	4	30	1999-5-1	Y1 M1 D3
4	1999	4	31	无效	Y1 M1 D4
5	1999	1	15	1999-1-16	Y1 M2 D1
6	1999	1	29	1999-1-30	Y1 M2 D2
7	1999	1	30	1999-1-31	Y1 M2 D3
8	1999	1	31	1999-2-1	Y1 M2 D4
9	1999	2	15	1999-2-16	Y1 M3 D1
10	1999	2	29	无效	Y1 M3 D2
11	1999	2	30	无效	Y1 M3 D3
12	1999	2	31	无效	Y1 M3 D4
13	1999	12	15	1999-12-16	Y1 M4 D1
14	1999	12	29	1999-12-30	Y1 M4 D2
15	1999	12	30	1999-12-31	Y1 M4 D3
16	1999	12	31	2000-1-1	Y1 M4 D4
17	2000	4	15	2000-4-16	Y2 M1 D1
18	2000	4	29	2000-4-30	Y2 M1 D2
19	2000	4	30	2000-5-1	Y2 M1 D3
20	2000	4	31	无效	Y2 M1 D4
21	2000	1	15	2000-1-16	Y2 M2 D1
22	2000	1	29	2000-1-30	Y2 M2 D2
23	2000	1	30	2000-1-31	Y2 M2 D3
24	2000	1	31	2000-2-1	Y2 M2 D4
25	2000	2	15	2000-2-16	Y2 M3 D1
26	2000	2	29	2000-3-1	Y2 M3 D2
27	2000	2	30	无效	Y2 M3 D3
28	2000	2	31	无效	Y2 M3 D4
29	2000	12	15	2000-12-16	Y2 M4 D1
30	2000	12	29	2000-12-30	Y2 M4 D2
31	2000	12	30	2000-12-31	Y2 M4 D3
32	2000	12	31	2001-1-1	Y2 M4 D4

非月末：变量 day+1 作为输出日期。

月末：变量 month+1、变量 day+1 作为输出日期。

年末：变量 year+1、变量 month=1、变量 day=1 作为输出日期。

表 5-4 中仍然存在不足。

首先，变量 day 的等价类 D1={day：1≤day≤28}划分没有考虑到闰年和平年 2 月份月末日期的不同和因此造成的输出处理差别。例如，1999 年 2 月 28 日是月末，NextDate 函数的输出是 1999-3-1；而 2000 年 2 月 28 不是月末，NextDate 函数的输出是 1999-2-29。

其次，变量 day 的等价类 D2={day：day=29}的划分仅对 2 月份的输出处理有意义，对于其他月份来说，D2 和 D1 的处理无差别，这造成了部分冗余。例如，表 5-4 中第 18 行和第 17 行一样是非月末，NextDate 函数的输出处理也是一样的，因此属于冗余。类似地，变量 day 的等价类 D3={day：day=30}的划分仅对 M1={month：month 有 30 天}的输出处理有意义，对 M2={month：month 有 31 天，除去 12 月}和 M4={month：month 是 12 月}来说，D3 和 D1 的处理无差别，这同样造成了部分冗余。例如，表 5-4 中第 23 行和第 22 行一样是非月末，NextDate 函数的输出处理也是一样的，属于冗余。

最后，对于 2 月份来说，NextDate 函数对变量 day 的等价类 D3={day：day=30}、D4={day：day=31}划分的输出处理也是一样的，也属于冗余，例如，表 5-4 中第 12 行和第 11 行一样是超过 2 月份日期范围的非法输入，NextDate 函数的输出处理也是一样的，属于冗余。

针对上述问题，我们增加非闰年和闰年 2 月 28 日的测试用例(第 33 和 34 行)，并删除掉表 5-4 中的 10 个冗余用例(第 2、6、7、14、15、18、22、23、30 和 31 行)，最终生成 24 个测试用例，结果如表 5-5 所示。

表 5-5　NextDate 函数强一般等价类用例设计优化

用例编号	输　入			预 期 输 出	覆盖有效等价类
	year	month	day		
1	1999	4	15	1999-4-16	Y1 M1 D1
3	1999	4	30	1999-5-1	Y1 M1 D3
4	1999	4	31	无效	Y1 M1 D4
5	1999	1	15	1999-1-16	Y1 M2 D1
8	1999	1	31	1999-2-1	Y1 M2 D4
9	1999	2	15	1999-2-16	Y1 M3 D1
10	1999	2	29	无效	Y1 M3 D2
11	1999	2	30	无效	Y1 M3 D3
12	1999	2	31	无效	Y1 M3 D4
13	1999	12	15	1999-12-16	Y1 M4 D1
16	1999	12	31	2000-1-1	Y1 M4 D4
17	2000	4	15	2000-4-16	Y2 M1 D1

续表

用例编号	输　入			预 期 输 出	覆盖有效等价类
	year	month	day		
19	2000	4	30	2000-5-1	Y2 M1 D3
20	2000	4	31	无效	Y2 M1 D4
21	2000	1	15	2000-1-16	Y2 M2 D1
24	2000	1	31	2000-2-1	Y2 M2 D4
25	2000	2	15	2000-2-16	Y2 M3 D1
26	2000	2	29	2000-3-1	Y2 M3 D2
27	2000	2	30	无效	Y2 M3 D3
28	2000	2	31	无效	Y2 M3 D4
29	2000	12	15	2000-12-16	Y2 M4 D1
32	2000	12	31	2001-1-1	Y2 M4 D4
33	1999	2	28	1999-3-1	Y1 M1 D1
34	2000	2	28	2000-2-29	Y2 M1 D1

5.3　边界值分析

边界值分析(Boundary Value Analysis)是一种通过选择等价类边界值设计测试用例的方法。边界值分析法不仅重视输入条件边界，而且必须考虑输出域边界。边界值分析方法是对等价类划分方法的补充。

长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入或输出范围的内部。因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例时，应首先确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。边界值分析方法通常选取正好等于、刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值作为测试数据。

一般来说，基于边界值分析方法选择测试用例的原则如下。

(1) 如果输入条件规定了值的范围，则应取刚达到这个范围的边界的值，以及刚刚超越这个范围边界的值作为测试输入数据。

例如，两位整数加法器数的范围是[－99，99]，则应测试边界数值－99，－98，－100 和 99，98，100。

(2) 如果输入条件规定了值的个数，则用最大个数、最小个数、比最小个数少 1、比最大个数多 1 的数作为测试数据。

例如，姓名要求包括 1～20 个字符，则需要测试姓名包括 1 个字符、0 个字符、2 个字符、20 个字符、19 个字符、21 个字符的情况作为边界值测试。