CHAPTER 3

第 3 章

动态测试技术

3.1 黑盒测试技术

3.1.1 边界值分析法

我们知道，函数可以理解为从一个集合（函数的定义域）值映射到另一个集合（函数的值域），定义域和值域可以是其他集合的叉积。任何程序都可以看作是一个函数，程序的输入构成函数的定义域，程序的输出构成函数的值域。定义域测试是著名的功能性测试方法之一。这种形式测试的重点是从输入变量的定义域来进行分析并设计出测试用例，但实际上，也可以根据被测程序本身的特点基于变量的值域来分析并设计测试用例。从定义域或值域来分析并设计测试用例往往能互相补充，其基本思想均源于函数。

1. 基本边界值分析

为了便于理解，先讨论具有两个变量 x_1 和 x_2 的函数 F。如果函数 F 对应一个程序，那么输入的两个变量 x_1 和 x_2 的值应该存在取值的边界，其边界值要根据程序的需求来确定，变量的边界值可能是显示的，也可能是隐含的，如果是隐含的，则需要根据实际情况进行分析。这里假设变量 x_1 和 x_2 有如下边界：

$$a \leqslant x_1 \leqslant b$$

$$c \leqslant x_2 \leqslant d$$

边界值分析关注的是输入变量的边界，依据边界来设计测试用例。边界值测试的基本原理是程序的错误或缺陷可能出现在输入变量的极限值附近。例如，程序中循环语句的循环次数可能会多一次或少一次，就涉及边界值问题；超市销售系统中的食品保质日期是一个边界值问题；银行系统每天的取款限额也是一个边界值问题。在我们的生活中，边界值问题比比皆是。

基本边界值分析的基本思想是在输入变量的取值区间内取最小值、略高于最小值、正常值、略低于最大值和最大值 5 个值。边界值分析也是基于一种关键假设，这种假设称“单缺陷”假设，即由于缺陷导致的程

序失效极少是由两个(或多个)缺陷的同时作用引起,也就是程序的失效极少是由于两个(或多个)变量在其边界值附近取值引起的,而是单个变量在其边界值附近取值引起的。

基本边界值分析的测试用例设计规则是:通过使所有变量取正常值,而使其中的一个变量取最小值(min),比最小值大的值(min+),位于或接近中间的正常值(nom),比最大值小的值(max-)和最大值(max)这 5 个值,每个变量分别取一次。下面是两个变量的基本边界值分析的测试用例的输入组合:

$$\{\langle x_{1\text{nom}}, x_{2\text{min}}\rangle, \langle x_{1\text{nom}}, x_{2\text{min+}}\rangle, \langle x_{1\text{nom}}, x_{2\text{nom}}\rangle, \langle x_{1\text{nom}}, x_{2\text{max-}}\rangle,$$
$$\langle x_{1\text{nom}}, x_{2max}\rangle, \langle x_{1\text{min}}, x_{2\text{nom}}\rangle, \langle x_{1\text{min+}}, x_{2\text{nom}}\rangle, \langle x_{1\text{nom}}, x_{2\text{nom}}\rangle,$$
$$\langle x_{1max\_}, x_{2\text{nom}}\rangle, \langle x_{1max}, x_{2\text{nom}}\rangle\}$$

以上为 10 个测试用例的输入,实际上只要考虑 9 个就可以了,因为当两个变量都取位于或接近中间的正常值(nom)情况时的测试用例有两个,这两个测试用例在实际的测试过程中的效果是相同的,一般不会有新发现。就程序的执行路径而言,这两个测试用例执行的路径相同,即也不会发现新错误或缺陷,因此可以省略其中之一。

那么对于 n 个变量的被测程序,基本边界值分析的测试用例数为:对于有 n 变量程序,每次使除一个以外的所有其他变量取正常值,使剩余的那个变量分别取最小值、略高于最小值、位于或接近中间的正常值、略低于最大值和最大值,对每个变量都重复进行一次。这样,对于一个 n 变量函数,基本边界值分析法会产生 $4n+1$ 个测试用例。

基本边界值分析法可以采用两个步骤:分析变量数和变量的值域。分析变量数,可以根据所测试的程序本身进行分析,例如,在机票定购系统中查询航班功能,输入的变量可能有出发地、目的地、出发时间、人数、时间段共 5 个变量。确定变量的值域取决于变量本身的性质,例如,对于万年历中的日期处理有月份(m)、天(d)和年(y)三个变量,对于变量 d 和变量 m,无论是定义成枚举类型还是其他数值类型均能很容易地确定其值域;而对于变量 y,可以根据所测试程序实际情况指定一个“人工”值域。值域确定后就可以根据变量的值域取最小值、略高于最小值、正常值、略低于最大值和最大值了。对于“人工”指定的值域要根据具体的情况去考虑,甚至可以取该变量类型允许的最大值和最小值。

边界值分析对布尔变量没有什么意义,布尔取值为 TRUE 和 FALSE,其余三个值不明确。布尔变量可以用后面论述的决策表方法进行测试。

逻辑变量也可以用“遍历”边界值分析来设计测试用例。例如在 ATM 例子中,银行业务处理类型是逻辑变量,其只有三个值:存款,取款和查询。密码也是一个逻辑量,假设进入某系统的密码为 4 位,那么“遍历”所有可能的组合则很困难。所以设计测试用例时根据情况决定。

基本边界值分析具有局限性。如果被测程序有多个独立的变量,这些变量也是物理量,则很适合用边界值分析。这里的关键词是“独立”和“物理量”。例如,万年历中的月份、日期和年三变量之间具有依赖关系,虽然三个变量具有物理量的性质,但边界值分析没有考虑到变量之间的依赖,这样用边界值分析法设计的测试用例其测试效果则不佳。物理量准则决定了物理量的实际含义,对用例的设计很重要。例如,变量(物理量)表示温度、压力、空气速度、迎角、负载等,则对于边界值分析极为重要。如监控系统监控的温度范围;医疗分析系统使用的步进电机确定要分析的样本传送带的位置等都是物理量的例子。物理量便于我们确定变量的值域。

2. 健壮性边界分析

健壮性边界分析是基本边界值分析的一种简单扩展。除了变量的 5 个边界值分析取值以外，还要取一个略超过最大值(max+)的值，以及取一个略小于最小值(min−)的值，以测试超过边界极值时系统会有什么表现。

基本边界值分析的大部分讨论都直接适用于健壮性边界分析。健壮性边界分析最有意思的部分不是输入，而是程序的预期输出。当物理量超过其最大值或小于其最小值时程序会出现什么情况呢？如果变量是代表飞机机翼的迎角，超出值域范围可能会使飞机失速；如果变量是代表电梯的负荷能力，当超出规定的重量时会出现什么情况？健壮性边界分析主要的价值是观察程序的例外处理情况。

健壮性边界分析的测试用例个数分析与基本边界值类似，其理论测试用例数为 $6n+1$，其中 n 为变量的个数。

3. 最坏情况边界分析

在基本边界值分析方法中，我们提及边界值测试分析采用了“单缺陷”假设。除了这种“单缺陷”假设之外，还有所谓的“多缺陷”假设的情况，也就是程序的失效是由于两个(或多个)变量值在其边界值附近取值共同引起的，而不是由单个变量在其边界值附近取值引起的。

当我们关心多个变量取极值时程序可能会出现失效的情况时，这在电子电路分析中叫做“最坏情况测试”，在这里也使用这种思想来讨论最坏情况的边界分析来设计测试用例。其方法是：对每个变量，首先取包含最小值、略高于最小值、正常值、略低于最大值和最大值 5 个值构成一个集合，然后对这些集合进行笛卡儿积计算，生成的新集合中的每个元素均是一个测试用例的输入。

对于两个变量 x_1 和 x_2 的情况如下：

$$A = \{x_{1\min}, x_{1\min+}, x_{1\mathrm{nom}}, x_{1\max-}, x_{1\max}\}$$

$$B = \{x_{2\min}, x_{2\min+}, x_{2\mathrm{nom}}, x_{2\max-}, x_{2\max}\}$$

$$A \times B = \{\langle x_{1\min}, x_{2\min}\rangle, \langle x_{1\min}, x_{2\min+}\rangle, \langle x_{1\min}, x_{2\mathrm{nom}}\rangle,$$
$$\langle x_{1\min}, x_{2\max-}\rangle, \langle x_{1\min}, x_{2\max}\rangle, \langle x_{1\min+}, x_{2\min}\rangle, \langle x_{1\min+}, x_{2\min+}\rangle,$$
$$\langle x_{1\min+}, x_{2\mathrm{nom}}\rangle, \langle x_{1\min+}, x_{2\max-}\rangle, \langle x_{1\min+}, x_{2\max}\rangle, \cdots\}。$$

笛卡儿积生成的新集合共有 25 个元素，故有 25 个测试用例集合。

集合 A 和 B 的笛卡儿积中的元素就是测试用例的输入。最坏情况测试显然更彻底，因为基本边界值分析的测试用例是最坏情况边界值分析测试用例集合的真子集。最坏情况测试还意味着花费更多的工作量，即 n 变量函数的最坏情况测试，会产生 5^n 个测试用例，n 为变量的个数。

从以上的分析中，看出诸如测试用例的输入组合$<x_{1\min+}, x_{2\min+}>$，这里 x_1 和 x_2 分别取了值域的最小值，这是“多缺陷”假设的体现。

最坏情况边界分析与基本边界值分析一样，两者也有相同的局限性，特别是独立性要求方面的局限性。最坏情况边界分析的最佳运用是物理变量本身存在大量交互的情况，或者在程序失效的代价极高的情况下采用。

除了上述方法之外，还有一种更为极端的边界值分析方法，即健壮最坏情况边界值分析。其测试用例的设计是对每个变量分别取比最小值小、最小值、略高于最小值、正常值、略低于最大值、最大值、比最大值大共 7 个值构成一个集合，然后对这些变量的取值集合进行笛卡儿积计算，生成的新集合中的每个元素均是一个测试用例的输入。使用健壮最坏情况边界分析的

测试用例个数为 7^n，n 为变量的个数。

4. 边界值分析设计测试用例的原则

用边界值分析设计测试用例应遵循以下几条原则：

(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。

(2) 如果输入变量规定了值的个数，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。

(3) 边界值分析同样适用于输出变量，根据规格说明的每个输出条件，使用前面的原则(1)和(2)。

(4) 如果程序的规格说明给出的输入域或输出域是有序集合，则应选取集合的第一个元素和最后一个元素来设计测试用例。

(5) 如果程序中使用了一个内部数据结构，则应当选择这个内部数据结构边界上的值来设计测试用例。

(6) 分析规格说明，找出其他可能的边界条件。

(7) 分析变量的独立性，以确定边界值分析法的合理性。

(8) 在取中间值或正常值时，只要取接近取值范围中间的值就可以了。

(9) 在取比最小值小的值时，根据情况可以取多个，可以取负值、0 和小数。

(10) 在取比最大值大的值时，根据情况可以取多个，当最大值非指定时，根据业务具体分析。

3.1.2 等价类测试法

使用等价类作为功能性测试的基础有两个方面考虑：希望所设计的测试用例比较完备，同时又避免测试用例的冗余。边界值测试方法不能很好地解决这两个方面的问题，即研究使用边界值分析法设计的测试用例，很容易看出测试用例存在大量冗余，再进一步仔细研究，还会发现测试用例的设计存在严重漏洞，其原因主要是没有考虑到同一个变量的多区间或多意性，也没有考虑到不同变量之间的依赖关系。等价类测试法从另外一种角度来设计测试用例，其用例设计也使用了“单缺陷假设”和“多缺陷假设”的思想。

1. 等价类的基本思想

在前面的关系概念中讨论过满足等价关系的元素构成等价类。等价类面临的问题是如何对变量(输入或输出变量)划分等价类，同时要分析和考虑等价类划分的粒度问题，根据变量划分成的等价类构成了不同的子集，这些子集的并即是变量的整个集合或全集。这对于测试用例的设计有两点非常重要的意义：子集并成整个集合或全集提供了测试用例设计的完备性；而子集之间的互不相交可保证测试用例设计的一种形式上的无冗余。由于子集是由等价关系决定的，因此子集内的所有元素或所有点在所研究的业务领域内具有共同的性质。等价类测试的思想是通过对每个等价类中取一个元素或一个点来作为测试用例，如果等价类划分合理，则可以大大降低测试用例数量和测试用例之间的冗余。例如，根据输入的三条边判断输出的三角形类型的例子中，应该设计一个测试用例，其输出结果是一个等边三角形的情况，这样的测试用例我们可能选择一个三元组(5.5,5.5,5.5)作为测试用例的输入，也可以选择诸如(6,6,6)和(100,100,100)这样的测试用例输入。直觉告诉我们，程序对这些测试用例的执行过程和第一个测试用例是相同的，因此，其他两个测试用例是冗余的。再如，对于具有不同账户类

型的银行系统进行测试也存在类似的等价类问题。如果结合白盒测试(结构性)来理解,会看到具有同样账务类型的测试用例在执行时程序的“处理”是相同的,映射到白盒测试去理解就是“遍历相同的执行路径”。

等价类测试的关键就是依据等价关系划分等价类。我们用一个简单的例子说明等价类的划分问题。为了便于理解,这里讨论一个有两个变量 x_1 和 x_2 的程序 P。输入变量 x_1 和 x_2 拥有以下边界以及边界内的区间:

$a \leqslant x_1 \leqslant e$,区间为$[a,b),[b,c),[c,d),[d,e]$

$f \leqslant x_2 \leqslant h$,区间为$[f,g),[g,h]$

其中方括号和圆括号分别表示闭区间和开区间的端点。x_1 和 x_2 的无效区间是 $x_1 \langle a, x_1 \rangle e$,以及 $x_2 \langle f, x_2 \rangle h$。如图 3-1 所示。

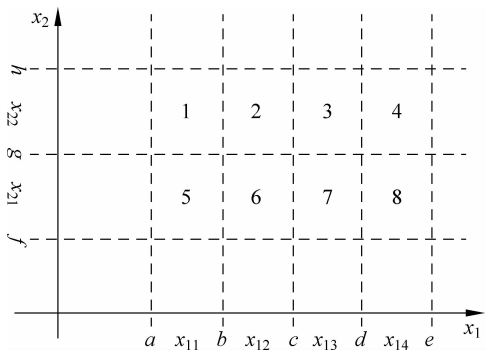

图 3-1 弱一般等价类测试用例分布

2. 弱一般等价类测试

上面的例子中两个变量 x_1 和 x_2,根据其范围作图 3-1 所示的标识分析,从图中可以看出标识为 1,2,3,4,5,6,7,8 的范围的区域均可以理解为一个有效等价类,因为在这些不同的区间内其所有的点具有同样的特性,即符合等价关系的定义,如在区间 1 中的所有点同时符合 $a<x_1<b, g<x_2<h$。

弱等价类测试用例的设计基于如下因素考虑:

- 基于单缺陷假设;
- 测试用例的个数是变量划分区间最多的那个变量的有效区间个数;
- 测试用例的选取应该考虑分布的均匀。

根据以上分析,对于有两个变量 x_1 和 x_2 的程序 P 的弱等价类测试用例的个数为 4 个,测试用例分布可以是图 3-1 中标号为 1,6,3,8 或者是标号为 5,2,7,4 的区域(有效等价类)的任一组。这组 x_1 和 x_2 的值的组合构成弱一般等价类的测试用例的输入。

3. 强一般等价类测试

强一般等价类测试与弱一般等价类测试的不同主要在于强等价类测试是基于多缺陷假设,需要从不同的输入或输出变量划分的有效等价类中或区间中取一个值分别构成集合,这些不同变量取值构成的集合的笛卡儿积中的每个元素就对应一个强一般等价类的测试用例的输入。下面是针对图 3-1 进行的分析。

变量 x_1 和 x_2 在其有效区间构成的集合是:

$A = \{x_{11}, x_{12}, x_{13}, x_{14}\}$

$B = \{x_{21}, x_{22}\}$

$A \times B = \{\langle x_{11}, x_{21} \rangle, \langle x_{11}, x_{22} \rangle, \langle x_{12}, x_{21} \rangle, \langle x_{12}, x_{22} \rangle, \langle x_{13}, x_{21} \rangle, \langle x_{13}, x_{22} \rangle, \langle x_{14}, x_{21} \rangle, \langle x_{14}, x_{22} \rangle\}$。

在 A 和 B 的笛卡儿积($A \times B$)中的每个元素均对应图 3-1 中的一个有效等价类区域,所以,其测试用例应该覆盖 1,2,…,8 这 8 个区域(等价类),在每个区域内任一个点构成一个测试用例的输入。这 8 个区域就是 x_1 和 x_2 这两个变量的划分构成的有效等价类。

强一般等价类测试具有一定的完备性:一是保证测试用例覆盖所有的有效等价类,二是输入或输出变量每个有效区间或每个有效等价类之间的每个组合均能取一个测试用例。

通过例子可以看到,“好的”等价类测试的关键是等价关系划分的选择,最好的情况是每个等价类内具有被“相同处理”的特性或等价类内的点在我们研究的业务领域内具有同样的性质。特别强调的是,等价类划分既可以基于输入变量进行,也可以基于输出变量进行。

4. 弱健壮等价类测试

弱健壮等价类测试是在弱一般等价类测试的基础上考虑了无效等价类的情况。测试用例的设计思想仍然是考虑了单缺陷假设。弱健壮等价类测试的等价类划分原则与前面的等价类方法相同。其测试用例由两个部分构成:

- 弱一般等价类部分的测试用例。
- 额外弱健壮部分的测试用例。对于 n 个变量而言,在这 n 个变量中每次取一个变量,分别取这个变量的所有可能的无效值和其他 $n-1$ 个变量取有效值组合来构成测试用例的输入,保证如此取法涉及每个变量,即每个变量取一次。

对于上面的例子,即有两个变量 x_1 和 x_2 的程序 P,如图 3-2 所示,其变量 x_1 和 x_2 的无效区间均为两个,即 $x_1>e$,$x_1<a$ 和 $x_2>h$,$x_2<f$。

图 3-2 中弱健壮等价类测试用例来自于以下区域,包含两个部分:弱一般的部分,即在 1,6,3,8 这 4 个区域内或在 5,2,7,4 这 4 个区域内分别取一个测试用例,加上弱健壮部分,即在 9,10,…,20 这 12 个区域内取一个测试用例,构成测试用例集的测试用例输入或输出组合。

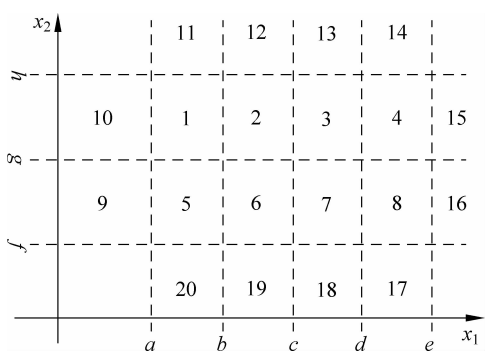

图 3-2 弱健壮等价类测试用例分布

健壮等价类测试主要测试输入或输出变量无效或例外的情况,在实际的测试中,需求规约中一般没有表达对无效或例外的处理,为无效等价类的测试用例设计带来不便,但测试人员应该积极地理解需求,努力划分可能的无效等价类,以找出程序对无效或例外情况处理的正确性。

5. 强健壮等价类测试

强健壮等价类测试是在强一般等价类测试的基础上考虑无效等价类的情况。测试用例的设计思想仍然考虑多缺陷假设。强健壮等价类测试的等价类划分原则与前面的等价类方法相同,其测试用例由两个部分构成:

- 强一般等价类部分的测试用例。
- 额外强健壮部分的测试用例。是在“额外弱健壮部分的测试用例”的基础上进一步考虑 n 个变量中两个或两个以上变量或所有变量都无效的情况下等价类内的取值。即在 n 个变量划分的等价类中(包含有效等价类和无效等价类)分别取值构成测试用例,在这些测试用例中 n 个变量的取值可以有一个变量取值来源于该变量的无效等价类,也可能存在其中的两个或多个或 n 个变量的取值全部来源于无效等价类。

实际上强健壮等价类的测试用例输入对应于每个变量区间取值(包括有效区间和无效区间)构成集合的笛卡儿积,笛卡儿积中的每个元素就是一个测试用例的输入组合。值得注意的是,这些元素可能是输入变量的组合,即是测试用例的输入;也可能是输出变量的组合,即是测试用例的输出。针对以上的例子,图 3-3 中标有数字的区域均是测试用例的取值区域,其中的 1,2,3,4,5,6,7,8 为强一般等价类部分的测试用例取值区域;9,10,11,12,13,14,15,16,

17,18,19,20,21,22,23,24 为额外强健壮部分的测试用例取值区域。

6. 等价类方法设计原则

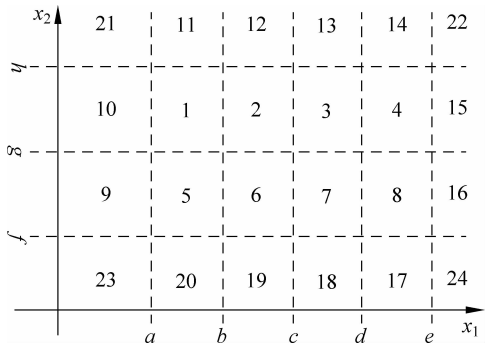

图 3-3 强健壮等价类测试用例分布

等价类是指某个输入域或输出域的子集合。在该子集合中,各个输入数据对于发现程序中的错误或缺陷都是等效的。因此,可以把全部输入或输出数据合理地划分为若干等价类,在每一个等价类中取一个数据作为测试的输入条件或输出条件,就可以用少量代表性的测试数据取得较好的测试结果。等价类划分有两种不同的情况:有效等价类和无效等价类。有效等价类是指对于程序的规格说明来说是合理的、有意义的输入数据或输出数据构成的集合,利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。无效等价类与有效等价类的定义恰巧相反。设计测试用例时,要同时考虑这两种等价类。因为软件不仅要能接收合理的数据,也要能经受意外的考验,这样的测试才能确保软件具有更高的可靠性。

下面给出 6 条确定等价类方法的设计原则:

- 在输入或输出条件规定了取值范围或取值个数的情况下,可以确立一个有效等价类和两个无效等价类。
- 在输入或输出条件规定了输入或输出值的集合或者规定了“必须如何”的条件的情况下,可以确立一个有效等价类和一个无效等价类。
- 在输入或输出条件是一个布尔量的情况下,可确定一个有效等价类和一个无效等价类。
- 在规定了输入数据的一组值(假定 n 个),并且程序要对每一个输入值分别处理的情况下,可确立 n 个有效等价类和一个无效等价类。
- 在规定了输入数据必须遵守的规则的情况下,可确立一个有效等价类(符合规则)和若干个无效等价类(从不同角度违反规则)。
- 在确知已划分的等价类中,各元素在程序中的处理方式的不同,则应再将该等价类进一步地划分为更小的等价类。
- 一个输入条件或一个输出条件均可能划分成多个有效等价类和多个无效等价类。

7. 举例

三角形问题:输入三角形的三条边 a、b、c,程序的输出是这三条边确定的三角形类型。如果 a、b 和 c 满足两边之和大于第三边,且三条边相等,则程序的输出是等边三角形。如果 a、b 和 c 满足两边之和大于第三边,且恰好有两条边相等,则程序的输出是等腰三角形。如果 a、b 和 c 满足两边之和大于第三边,且没有两条边相等,则程序输出的是不等边三角形。如果 a、b 和 c 不满足两边之和大于第三边,则程序输出的是非三角形。

(1) 等价类设计方法一:根据输入变量划分等价类。

根据输入变量划分等价类主要是等价类划分的粒度问题,划分的粒度大了,测试用例的设计会有漏洞;粒度太小,则测试用例会有冗余。

第一次尝试,将等价类作如下划分:

D1={〈a,b,c〉: a=b=c}

D2={⟨a,b,c⟩：a=b,a≠c}

D3={⟨a,b,c⟩：a=c,a≠b}

D4={⟨a,b,c⟩：b=c,a≠b}

D5={⟨a,b,c⟩：a≠b,a≠c,b≠c,且任意两边之和大于第三边}

分析一下这样的划分，可以看出等价类 D2,D3,D4 的划分粒度过大，这些等价类可能包括构成三角形和不构成三角形的情况，设计出的测试用例会有漏洞。所以必须进行第二次划分尝试，将 D2,D3 和 D4 分别划分成 D21,D22、D31,D32 及 D41,D42。具体划分的结果如下：

D1={⟨a,b,c⟩：a=b=c}

D21={⟨a,b,c⟩：a=b,a≠c,a+b>c}

D22={⟨a,b,c⟩：a=b,a≠c,a+b≤c}

D31={⟨a,b,c⟩：a=c,a≠b,a+c>b}

D32={⟨a,b,c⟩：a=c,a≠b,a+c≤b}

D41={⟨a,b,c⟩：b=c,a≠b,b+c>a}

D42={⟨a,b,c⟩：b=c,a≠b,b+c≤a}

D5={⟨a,b,c⟩：a≠b,a≠c,b≠c,且任意两边之和大于第三边}

根据以上划分结果，可以得出表 3-1 所示的测试用例，这里可以不考虑弱和强的情况，也不考虑 a,b 和 c 为负数和 0 的情况。

表 3-1 依据第二次尝试划分的测试用例

编 号	输入 a	输入 b	输入 c	对应的等价类	预 期 输 出
T1(D1)	10.5	10.5	10.5	D1	等边三角形
T2(D21)	20.11	20.11	30	D21	等腰三角形
T2(D22)	20	10	8	D22	非三角形
T4(D31)	2000	1500	1500	D31	等腰三角形
T5(D32)	33	80	33	D32	非三角形
T6(D41)	60	35	35	D41	等腰三角形
T7(D42)	90	40	40	D42	非三角形
T8(D5)	11	7	17	D5	不等边三角形

第二次等价类划分尝试后应该是基本合理的。但是还可以进行第三次划分尝试，即将 D22,D32 和 D42 划分成：

D221={⟨a,b,c⟩：a=b,a≠c,a+b<c}

D222={⟨a,b,c⟩：a=b,a≠c,a+b=c }

D321={⟨a,b,c⟩：a=c,a≠b,a+c<b}

D322={⟨a,b,c⟩：a=c,a≠b,a+c=b }

D421={⟨a,b,c⟩：b=c,a≠b,b+c<a}

D422={⟨a,b,c⟩：b=c,a≠b,b+c=a}

等价类这样划分的结果更加合理，将非三角形情况中的两边之和小于等于第三边分为两边之和小于第三边和等于第三边两种情况，其测试用例的个数达到了 11 个，具体测试用例这里略。

另外，在实际测试用例的设计时应该考虑 a、b 和 c 的值为无效情况的等价类或用边界值的方法设计一些测试用例做补充，即 a、b 和 c 的值为负值和 0 的情况。

(2) 等价类设计方法二：根据输出变量或输出的结果划分等价类。

根据三角形 4 种可能出现的输出：非三角形、不等边三角形、等腰三角形和等边三角形，设计如下的输出(值域)等价类：

R1={⟨a,b,c⟩：有三条边 a、b 和 c 的等边三角形}

R2={⟨a,b,c⟩：有三条边 a、b 和 c 的等腰三角形}

R3={⟨a,b,c⟩：有三条边 a、b 和 c 的不等边三角形}

R4={⟨a,b,c⟩：三条边 a、b 和 c 不构成三角形}

按照弱一般等价类测试用例的设计思想，弱一般等价类共有 4 个测试用例，与强一般等价类的测试用例个数相同。如表 3-2 所示。

表 3-2 三角形程序的弱一般和强一般测试用例

用例编号	a	b	c	预期输出
R1	6	6	6	等边三角形
R2	3.3	3.3	4.4	等腰三角形
R3	7	8	9	不等边三角形
R4	11	5	5	非三角形

虽然等价类的划分是以输出结果进行的，但是在考虑等价类的健壮情况时，还是要从输入变量入手。这里为了便于问题的讨论，假设 a、b、c 的范围如下：0＜a＜800，0＜b＜800，0＜c＜800。那么根据弱健壮等价类设计思想，三角形问题的额外弱健壮等价类测试用例部分如表 3-3 所示，当然还应该考虑 a、b 和 c 为 0 的情况。

表 3-3 额外弱健壮测试用例

用例编号	a	b	c	预期输出
W1	－1	22	25.5	a 取值不在范围之内
W2	801	3.3	3.3	a 取值不在范围之内
W3	0	5.4	8.0	a 不能为 0
W4	15	－2	9	b 取值不在范围之内
W5	11	803	5	b 取值不在范围之内
W6	15	0	10.3	b 不能为 0
W7	10	2.5	－5	c 取值不在范围之内
W8	19.3	790	880	c 取值不在范围之内
W9	25.1	29	0	c 不能为 0

根据额外强健壮等价类设计思想，变量 a、b、c 中可能有一个无效，也可能有两个无效，也甚至三个全都无效。基于数学的组合知识得到额外的强健壮等价类测试用例数为 $c_3^1c_2^1+c_3^2c_2^1c_2^1+c_3^3c_2^1c_2^1c_2^1=26$，这里没考虑 a、b 和 c 为 0 的情况。表 3-4 是部分额外的强健壮等价类测试用例。

表 3-4 部分额外的强健壮等价类测试用例

用例编号	a	b	c	预 期 输 出
SW1	－1	22	25.5	a 取值不能为负
SW2	3.3	801	3.3	b 取值不在范围之内
SW3	15	9	－2	c 取值不能为负
SW4	－5.3	803	5	a 不能为负，b 取值不在范围之内

续表

用例编号	a	b	c	预 期 输 出
SW5	10	−2.5	−7	b,c 不能为负
SW6	−19.3	799	807	a 不能为负,c 取值不在范围之内
SW7	−1.3	−2.7	−0.01	a,b,c 取值不能为负
SW8	809	−3.33	−4	a 取值不在范围之内,b、c 不能为负

通过上面例子的第二种设计方法,得到三角形程序按照输出变量划分等价类的测试用例结果:

- 弱一般等价类测试用例数:4。
- 强一般等价类测试用例数:4。
- 弱健壮等价类测试用例数:13。
- 强健壮等价类测试用例数:30(不考虑 a、b 和 c 为 0 的情况)。如果考虑 a、b 和 c 为 0 的情况,测试用例数为 121($c_3^1c_3^1+c_3^2c_3^1c_3^1+c_3^3c_3^1c_3^1c_3^1=117+4$)。

值得进一步思考的是,对于所测程序,如果每个变量(输入或输出)均能划分成不同的等价类,其弱、强等价类方法设计测试用例的思路是一致的,重要的是要灵活掌握。例如,万年历程序中的变量 year、month 和 day 就可以分别划分成不同的等价类。

3.1.3 错误推测法

错误推测法就是基于经验和直觉推测程序中所有可能存在的各种错误或缺陷,有针对性地设计测试用例的方法。

错误推测法的基本思想是列举出程序中所有可能有的错误或缺陷和容易发生错误或缺陷的特殊情况,根据这些推测来设计测试用例。错误推测法本身不是一种测试技术,而是一种可以应用到所有测试技术中产生更加有效测试的一种技能,例如,设计一些非法、错误、不正确和垃圾数据进行输入测试是很有意义的。如果软件要求输入数字,就输入字母。如果软件只接受正数,就输入负数。如果软件对时间敏感,就输入异常的时间,如在公元 3000 年是否还能正常工作。再如,在单元测试时曾发现并列出的许多在模块中经常出现的错误,以前软件测试中曾经发现的错误等,这些就是经验的总结。另外,错误推测法常常会考虑输入数据和输出数据为 0 的情况,或者输入表格为空格或输入表格只有一行,这些都是容易发生错误的情况,可根据这些情况设计测试用例。

总之,用好错误推测法应该具备如下条件:

- 充分地理解业务;
- 具有开发和测试的实际经验;
- 掌握全面的测试技术。

下面是三个具体的例子。

(1) 测试两位加法计算器中错误推测法的测试用例举例,如表 3-5 所示。

(2) 对一个排序程序,根据推测可以列出可能存在错误或缺陷的几种情况:

- 输入表为空。
- 输入表中只有一行。
- 输入表中所有的行都具有相同的值。
- 输入表已经是排序的。