**黑盒测试**

[● 3黑盒测试 2](#_Toc2334)

[● 3.1对程序的功能性测试要求 2](#_Toc27146)

[● 3.2优点 3](#_Toc30040)

[● 3.3缺点 3](#_Toc8424)

[● 3.4意义 3](#_Toc28347)

[● 3.5原则 3](#_Toc10473)

[● 3.6方法 4](#_Toc4064)

[● 等价类划分法 4](#_Toc22827)

[● 定义 4](#_Toc1704)

[● 原则 4](#_Toc5699)

[● 边界值分析法 5](#_Toc17069)

[● 定义 5](#_Toc368)

[● 原则 5](#_Toc14351)

[● 组合测试用例设计技术 6](#_Toc16044)

[● 因果图法 6](#_Toc25045)

[● 定义 6](#_Toc28685)

[● 因果图 6](#_Toc18494)

[● 特点 7](#_Toc14666)

[● 导出测试用例 7](#_Toc12984)

[● 判定表驱动测试法 7](#_Toc29803)

[● 定义 7](#_Toc10830)

[● 判定表的组成 8](#_Toc18661)

[● 规则合并 8](#_Toc22734)

[● 建立步骤 9](#_Toc8375)

[● 优缺点 9](#_Toc29186)

[● 场景分析法 9](#_Toc4926)

[● 定义 9](#_Toc32675)

[● 如何使用 10](#_Toc23514)

[● 功能图法 10](#_Toc28145)

[● 错误推测法 10](#_Toc25250)

[● 特殊值测试 11](#_Toc26143)

[● 故障猜测法 11](#_Toc20396)

[● 随机测试 11](#_Toc11496)

[● 常用控件测试 11](#_Toc9662)

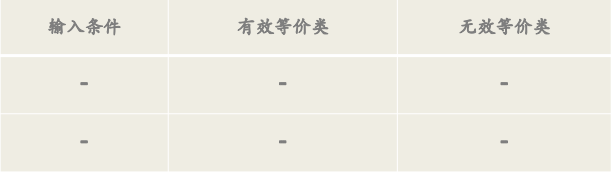
[● 文本框测试 11](#_Toc12043)

[● 按钮测试 11](#_Toc1892)

[● 单选按钮测试 11](#_Toc29322)

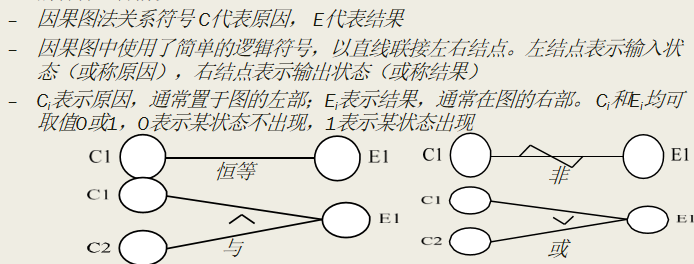
[● 复选框测试 11](#_Toc8629)

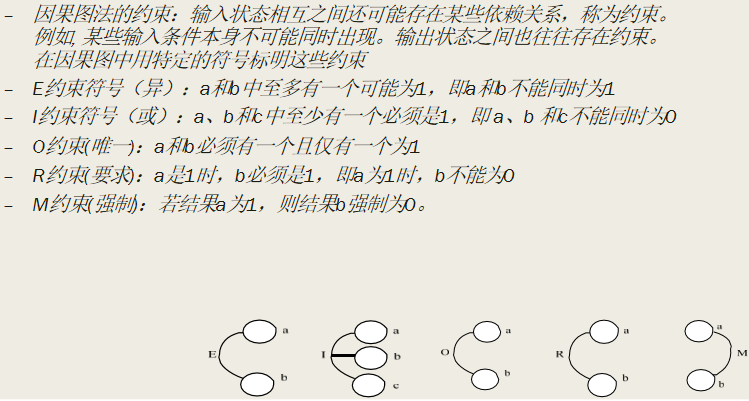
[● 方法选择的综合策略 11](#_Toc6865)

1. **3黑盒测试**
   1. **3.1对程序的功能性测试要求**
      1. 每个软件特性必须被一个测试用例或一个被认可的异常所覆盖
      2. 用数据类型和数据值的最小集测试
   2. **3.2优点**
      1. 黑盒测试与软件具体实现无关，所以如果软件实现发生了变化，测试用例仍然可以使用
      2. 设计黑盒测试用例可以和软件实现同时进行，因此可以压缩项目总的开发时间
      3. 从产品功能角度测试可以最大程度满足用户的需求
      4. 相同动作可重复执行，最枯燥的部分可由机器完成
      5. 依据测试用例针对性地找寻问题，定位更为准确，容易生成测试数据
      6. 将测试直接和程序/系统要完成的操作相关联
   3. **3.3缺点**
      1. 代码得不到测试
      2. 如果规格说明设计有误，很难发现
      3. 测试不能充分的进行
      4. 结果取决于测试用例的设计
   4. **3.4意义**
      1. 黑盒测试有助于对被测产品进行总体功能的需求进行验证
      2. 从测试管理来说，黑盒测试是非常方便的，不需要对代码进行测试管理
      3. 黑盒测试是把所有可能的输入都作为测试数据使用的，容易查出程序中的错误
   5. **3.5原则**
      1. 根据软件规格说明书设计测试用例，规格说明书的正确性是至关重要的
      2. 有针对性的地找问题，并且正确定位等价类
      3. 功能是否有缺陷或错误现象？
      4. 根据测试的重要性来确定测试等级和测试重点，减少程序可能出现的缺陷
      5. 在接口处，输入的信息是否能正确接受？接受后能否输出正确的结果
      6. 认真选择测试策略，尽可能发现程序的数据结构错误或外部信息访问错误，站在用户立场上进行测试
   6. **3.6方法**
      1. **等价类划分法**
         1. **定义**
            1. 一种典型的、重要的黑盒测试方法，它将程序所有可能的输入数据（有效的 和无效的）划分成若干个等价类。然后从每个部分中选取具有代表性的数据 当作测试用例进行合理的分类，测试用例由有效等价类和无效等价类的代表 组成，从而保证测试用例具有完整性和代表性。等价类划分为有效等价类和无效等价类
         2. **原则**
            1. 如果输入条件规定了取值范围，或值的个数，则可确立一个有效等价类和两个无效等价类。
            2. 如果输入条件规定了输入值的集合，或者是规定了“必须如何”的条件， 这时可确立一个有效等价类和一个无效等价类。例如，在Pascal语言中对变量标识符规定为“以字母打头的……串”。
            3. 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等 价类。
            4. 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理。这 时可为每一个输入值确立一个有效等价类，此外针对这组值确立一个无效等价类，它是所有不允许的输入值的集合。例如，在教师上岗方案中规定对教授、副教授、讲师和助教分别计算分数，做相 应的处理。因此可以确定4个有效等价类为教授、副教授、讲师和助教，一个无效等价类， 它是所有不符合以上身分的人员的输入值的集合
            5. 如果规定了输入数据必须遵守的规则，则可以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）。

例如，Pascal语言规定 “一个语句必须以分号‘;’结束”。这时可以确定一个有 效等价类“以‘;’结束”，若干个无效等价类“以‘:’结束”、“以‘,’结束”、 “以‘ ’结束”、“以LF结束”。

* + 1. **边界值分析法**
       1. **定义**
          1. 用于对输入或输出的边界值进行测试的一种典型、重要的黑盒测试方法。在测试过程中，边界值分析法是作为对等价类划分法的补充，专注于每个等价类的边界值，区别是在等价类中随机选取一个测试点
       2. **原则**
          1. 如果输入条件规定了值的范围，则应取刚刚到达这个范围边界的值，以及刚刚超过这个范围边界的值作为测试输入数据
          2. 如果输入条件规定了值的个数，则应取最大个数、最小个数、比最大个数多1， 比最小个数少1的数作为测试输入数据。
          3. 根据规格说明和每个输出条件，使用原则(1)（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（如有序表），则选取集 合的第一个元素和最后一个元素作为测试用例。
          5. 如果程序中使用了一个内部数据结构，则应选择此数据结构的边界上的值作为测 试用例。
    2. **组合测试用例设计技术**
    3. **因果图法**
       1. **定义**
          1. 一种适合于描述对于多种输入条件组合的测试方法，根据输入条件的组合、 约束关系和输出条件的因果关系，分析输入条件的各种组合情况，从而设计测试用例的方法，它适合于检查程序输入条件到输出结果的各种组合情况。 因果图法一般和判定表结合使用通过映射同时发生相互影响的多个输入来确 定判定条件。因果图法最终生成的就是判定表. 它适合于检查程序输入条件 的各种组合情况
       2. **因果图**





* + - 1. **特点**
         1. 考虑输入条件的组合关系
         2. 考虑输出条件对输入条件的信赖关系，即因果关系
         3. 测试用例发现错误的效率高
         4. 可检查出功能说明书中的某些不一致或遗漏
         5. 适合于检查程序输入条件和各种组合情况
      2. **导出测试用例**
         1. 分析软件规格说明的描述中那些是原因，那些是结果。原因是输入或输入条件的 等价类，结果是输出条件。并给每个原因和结果赋予一个标示符，根据这些关系， 画出因果图。
         2. 因果图上用一些记号表明约束条件或限制条件
         3. 对需求加以分析并把它们表示为因果图之间的关系图
         4. 把因果图转换成判定表
         5. 根据判定表的每一列作为依据，设计测试用例
    1. **判定表驱动测试法**
       1. **定义**
          1. 定表测试严格，能够将复杂的逻辑关系和多种条件组合的情况表达得既具 体又明确。针对不同的逻辑条件组合值，分别执行不同的操作。因此，使用 判定表能够设计书完整的测试用例集合。判定表一种针对存在条件、动作关 系或者因果关系的特性测试的用例设计方法
       2. **判定表的组成**
          1. 条件桩（Condition Stub）：列出了问题的所有条件.列出条件的次序没有约束
          2. 动作桩（Action Stub）：列出问题规定可能采取的操作.这些操作的排列顺序无关紧要
          3. 条件项（Condition Entry）：列出条件桩给出的条件并列 出所有可能取值。针对条件桩的取值在所有可能情况下的真假值
          4. 动作项（Action Entry）：列出在条件项的各种取值情况 下应该采取的动作
          5. 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。在判定表中贯穿条件项和动作项的一列 就是一条规则。有n个条件，每个条件有两个取值（0， 1），故有2n条规则。显然，判定表中列出多少组条件 取值，也就有多少条规则
       3. **规则合并**
          1. 就是有两条或多条规则合并为一条规则
          2. 左端：1，1 ；合并为右端：1
          3. 左端：－，0 ；合并为右端：－
          4. 左端：0，0 ；合并为右端：0
          5. 左端：1， 0；合并为右端：1
          6. 左端：0，1 ；合并为右端：1
       4. **建立步骤**
          1. 确定规则的个数加入有n个条件的决策表有2的n次方个规则（每个条件取真、 假值）
          2. 列出所有的条件桩和动作桩
          3. 填入条件项
          4. 填入动作项，得到初始判定表
          5. 简化判定表，合并相似规则
       5. **优缺点**
          1. 优点：在于它能够把复杂的问题按各种可能的情况一一列举出来,简明而易于理解, 也可避免遗漏。在一些数据处理问题当中，某些操作的实施依赖于多个逻辑 条件的组合，即：针对不同逻辑条件的组合值，分别执行不同的操作。判定 表很适合于处理这类问题
          2. 缺点：在于它不能表达重复执行的动作,例如循环结构
    2. **场景分析法**
       1. **定义**
          1. 一般把场景分为基本流和备选流。从一个流程开始，通过描述经过的路径来确定 业务过程，经过遍历所有的基本流和备选流来完成整个场景。基本流和备选流往 往是产品被同一事件的不同触发顺序和处理结果而形成的事件流，事件触发时的 情景便形成了场景。通过描述事件的不同触发情景，能够更好的使测试人员理解 场景，并设计出合适的测试用例，使测试用例更容易理解和执行。场景用来描述流经用例的路径，从用例开始到结束遍历这条路径上所有的基本流 和备选流
       2. **如何使用**
          1. 根据说明，描述出程序的基本流及各项备选流
          2. 根据基本流和各项备选流生成不同的场景
          3. 对每一个场景生成相应的测试用例
          4. 对生成的所有测试用例重新审查，去掉多余的测试用例， 确定测试用例后，为每一个测试用例确定测试数据值
    3. **功能图法**
       1. 用功能图形象地表示程序的功能说明，由状态迁移图和布尔函数组成，同时需要依靠判定表或因果图表示的逻辑功能，并机械地生成功能图的测试用例，功能图方法是黑盒、白盒混合用例的设计方法
       2. 功能图模型由状态迁移图和逻辑功能模型两部分构成，程序功能说明包括动 态说明和静态说明
       3. 状态迁移图：用于表示输入数据序列以及相应的输出数据；由输入数据和当 前状态决定输出数据和后续状态
       4. 逻辑功能模型：用于表示在状态中输入条件和输出条件的对应关系，由输入 数据决定输出数据，此模型只适用于描述静态说明。输出数据由输入数据决 定
       5. 动态说明：描述输入数据的次序或转移次序
       6. 静态说明：描述输入条件和输出条件之间的对应关系
    4. **错误推测法**
       1. 基于以往的经验和直觉，参照以往的软件系统出现的错误，推测程序中所有可能存在的各种缺陷和错误，从而有针对性地设计测试用例
       2. 基本思路是：列举出程序中所有可能的错误和容易发生错误的特殊情况，根 据可能出现的错误情况选择测试用例
    5. **特殊值测试**
       1. 特殊值测试就是指定软件中某些特殊值为测试用例而对软件实施的测试。这 些特殊值并不是根据某种方法推导出来的，而是根据测试人员的知识和经验得到的。通常情况下，特殊值测试人员都会从过去发生过的失效的事件，或者总会出现问题的情况，或者对于用户来说十分重要的事件中寻找特殊值
    6. **故障猜测法**
       1. 据经验和直觉猜测软件中可能存在的各种故障，从而有针对性地编写测试 这些故障的测试用例
    7. **随机测试**
       1. 对于给定的被测软件系统和软件系统的定义域，按照定义域中样本取值的概率，随机的选择其样本并作为其测试数据的过程称为随机测试
    8. **常用控件测试**
       1. **文本框测试**
       2. **按钮测试**
       3. **单选按钮测试**
       4. **复选框测试**
    9. **方法选择的综合策略**
       1. 对于业务流程清晰的被测系统，可以利用场景法贯穿整个测试案例过程，对主要业务流程进行测试。
       2. 当主要流程测完后，再对系统中的重要功能进行等价类划分测试，将无限测试变成有限测试，这是减少工作量和提高测试效率最有效的办法。
       3. 在任何情况下都必须使用边界值分析法。经验表明，用这种方法设计的测试用例发现程序错误的能力最强。
       4. 可以依靠测试工程师的智慧和经验追加一些测试用例。
       5. 对照程序逻辑，检查已设计的测试用例的逻辑覆盖程度。如果没有达到要求的覆盖标准|应当再补充足够的测试用例。
       6. 对于参数配置类的软件功能，要用正交试验法选择较少的组合方式达到最佳效果。