# 黑盒测试

黑盒测试(black—box testing)又称功能测试、数据[驱动](http://driver.it168.com/)测试或基于规范的测试。用这种方法进行测试时，被测程序被当作看不见内部的黑盒。在完全不考虑程序内部结构和内部特性的情况下，测试者仅依据程序功能的需求规范考虑确定测试用例和推断测试结果的正确性。因此黑盒测试是从用户观点出发的测试，黑盒测试直观的想法就是既然程序被规定做某些事，那我们就看看它是不是在任何情况下都做的对。完整的“任何情况”是无法验证的，为此黑盒测试也有一套产生测试用例的方法，以产生有限的测试用例而覆盖足够多的“任何情况”。由于黑盒测试不需要了解程序内部结构，所以许多高层的测试如确认测试、系统测试、验收测试都采用黑盒测试。  
    黑盒测试首先是程序通常的功能性测试。要求：  
    每个[软件](http://software.it168.com/)特性必须被一个测试用例或一个被认可的异常所覆盖；用数据类型和数据值的最小集测试；用一系列真实的数据类型和数据值运行，测试超负荷、饱和及其他“最坏情况”的结果；用假想的数据类型和数据值运行，测试排斥不规则输入的能力；对影响性能的关键模块，如基本算法、应测试单元性能(包括精度、时间、容量等)。  
    不仅要考核“程序是否做了该做的?”还要考察“程序是否没做不该做的2”同时还要考察程序在其他一些情况下是否正常。这些情况包括数据类型和数据值的异常等等。下述几种方法：(a)等价类划分，(b)因果图方法，(c)边值分析法，(d)猜错法，(e)随机数法，就是从更广泛的角度来进行黑盒测试。每一个方法都力图能涵盖更多的“任何情况”，但又各有长处，综合使用这些方法，会得到一个较好的测试用例集。

## 作用

[黑盒测试法](https://baike.baidu.com/item/%E9%BB%91%E7%9B%92%E6%B5%8B%E8%AF%95%E6%B3%95)注重于测试软件的功能需求，主要试图发现下列几类错误。

功能不正确或遗漏；

界面错误；

输入和输出错误；

[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)访问错误；

性能错误；

[初始化](https://baike.baidu.com/item/%E5%88%9D%E5%A7%8B%E5%8C%96)和[终止](https://baike.baidu.com/item/%E7%BB%88%E6%AD%A2)错误等。[1]

## 测试方法

### 概述

从理论上讲，黑盒测试只有采用穷举输入测试，把所有可能的输入都作为测试情况考虑，才能查出[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中所有的错误。实际上测试情况有无穷多个，人们不仅要测试所有合法的输入，而且还要对那些不合法但可能的输入进行测试。这样看来，完全测试是不可能的，所以我们要进行有针对性的测试，通过制定测试案例指导测试的实施，保证[软件测试](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95)有组织、按步骤，以及有计划地进行。黑盒测试行为必须能够加以量化，才能真正保证[软件质量](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E8%B4%A8%E9%87%8F)，而[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)就是将测试行为具体量化的方法之一。具体的黑盒[测试用例设计](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B%E8%AE%BE%E8%AE%A1)方法包括等价类划分法、边界值分析法、错误推测法、[因果图法](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE%E6%B3%95)、判定[表驱动](https://baike.baidu.com/item/%E8%A1%A8%E9%A9%B1%E5%8A%A8)法、正交试验设计法、功能图法、[场景](https://baike.baidu.com/item/%E5%9C%BA%E6%99%AF)法等。

等价类划分的办法是把[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)的输入域划分成若干部分（子集），然后从每个部分中选取少数代表性数据作为测试[用例](https://baike.baidu.com/item/%E7%94%A8%E4%BE%8B)。每一类的代表性数据在测试中的作用等价于这一类中的其他值。该方法是一种重要的，常用的黑盒[测试用例设计](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B%E8%AE%BE%E8%AE%A1)方法。

### 划分等价类

1) 划分等价类： 等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)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它值的测试.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.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。

有效等价类：是指对于[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)的规格说明来说是合理的，有意义的输入数据构成的集合.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](https://baike.baidu.com/item/%E6%97%A0%E6%95%88%E7%AD%89%E4%BB%B7%E7%B1%BB)：与有效等价类的定义恰巧相反。

设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)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。

### 划分等价类

2）划分等价类的方法：下面给出六条确定等价类的原则。

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。

②在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可确立一个有效等价类和一个[无效等价类](https://baike.baidu.com/item/%E6%97%A0%E6%95%88%E7%AD%89%E4%BB%B7%E7%B1%BB).

③在输入条件是一个[布尔量](https://baike.baidu.com/item/%E5%B8%83%E5%B0%94%E9%87%8F)的情况下，可确定一个有效等价类和一个无效等价类。

④在规定了输入数据的一组值（假定n个），并且[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)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。

⑤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](https://baike.baidu.com/item/%E6%97%A0%E6%95%88%E7%AD%89%E4%BB%B7%E7%B1%BB)（从不同角度违反规则）。

⑥在确知已划分的等价类中各元素在[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)处理中的方式不同的情况下，则应再将该等价类进一步的划分为更小的等价类。

3）设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)：在确立了等价类后，可建立等价类表，列出所有划分出的等价类：

### 输入条件

输入条件 有效等价类 [无效等价类](https://baike.baidu.com/item/%E6%97%A0%E6%95%88%E7%AD%89%E4%BB%B7%E7%B1%BB)

然后从划分出的等价类中按以下三个原则设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)：

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](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，使其尽可能多地覆盖尚未被覆盖地有效等价类，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](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，使其仅覆盖一个尚未被覆盖的[无效等价类](https://baike.baidu.com/item/%E6%97%A0%E6%95%88%E7%AD%89%E4%BB%B7%E7%B1%BB)，重复这一步.直到所有的无效等价类都被覆盖为止。

### 边界值分析法

边界值分析是通过选择等价类边界的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。边界值分析法不仅重视输入条件边界，而且也必须考虑输出域边界。它是对等价类划分方法的补充。

（1）边界值分析方法的考虑：

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，而不是发生在输入输出范围的内部.因此针对各种边界情况设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，可以查出更多的错误。

使用边界值分析方法设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，首先应确定边界情况.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.应当选取正好等于，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。

（2）基于边界值分析方法选择[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)的原则：

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，比最小个数少一，比最大个数多一的数作为测试数据。

3）根据规格说明的每个输出条件，使用前面的原则1）。

4）根据规格说明的每个输出条件，应用前面的原则2）。

5）如果[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)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作为[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。

6）如果[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中使用了一个内部[数据结构](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84)，则应当选择这个内部数据结构的边界上的值作为[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。

7）分析规格说明，找出其它可能的边界条件。

### 错误推测法

错误推测法是基于经验和直觉推测[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中所有可能存在的各种错误，从而有针对性的设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)的方法.

错误推测方法的基本思想： 列举出[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)中所有可能有的错误和容易发生错误的特殊情况，根据他们选择[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。 例如，在[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95)时曾列出的许多在模块中常见的错误. 以前产品测试中曾经发现的错误等，这些就是经验的总结。还有，输入数据和输出数据为0的情况. 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。可选择这些情况下的例子作为测试用例。

### 因果图法

前面介绍的等价类划分方法和边界值分析方法，都是着重考虑输入条件，但未考虑输入条件之间的联系，相互组合等。 考虑输入条件之间的相互组合，可能会产生一些新的情况. 但要检查输入条件的组合不是一件容易的事情，即使把所有输入条件划分成等价类，他们之间的组合情况也相当多. 因此必须考虑采用一种适合于描述对于多种条件的组合，相应产生多个动作的形式来考虑设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B). 这就需要利用[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)（逻辑模型）。

[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)方法最终生成的就是判定表。它适合于检查[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)输入条件的各种组合情况。

生成[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)

（1) 分析软件规格说明描述中，哪些是原因（即输入条件或输入条件的等价类），哪些是结果（即输出条件），并给每个原因和结果赋予一个标识符。

(2) 分析软件规格说明描述中的语义。找出原因与结果之间，原因与原因之间对应的关系. 根据这些关系，画出[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)。

(3) 由于语法或环境限制，有些原因与原因之间，原因与结果之间的组合情况不可能出现. 为表明这些特殊情况，在[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)上用一些记号标明约束或限制条件。

(4) 把[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)转换为[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)。

(5) 把[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)的每一列拿出来作为依据，设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)。

从[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)生成的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)（局部，组合关系下的）包括了所有输入数据的取TRUE与取FALSE的情况，构成的测试用例数目达到最少，且测试用例数目随输入数据数目的增加而线性地增加。

前面[因果图](https://baike.baidu.com/item/%E5%9B%A0%E6%9E%9C%E5%9B%BE)方法中已经用到了[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)。判定表（Decision Table）是分析和表达多逻辑条件下执行不同操作的情况下的工具.在[程序设计](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1)发展的初期，判定表就已被当作编写程序的[辅助工具](https://baike.baidu.com/item/%E8%BE%85%E5%8A%A9%E5%B7%A5%E5%85%B7)了.由于它可以把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。

### 判定表组成法

条件桩（Condition Stub）：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（Action Stub）：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（Condition Entry）：列出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（Action Entry）：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.在[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)中贯穿条件项和动作项的一列就是一条规则.显然，判定表中列出多少组条件取值，也就有多少条规则，既条件项和动作项有多少列。

判定表的建立步骤

①确定规则的个数。假如有n个条件.每个条件有两个取值（0,1），故有2n种规则。

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.等到初始判定表。

⑤简化.合并相似规则（相同动作）。

B. Beizer 指出了适合使用[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)设计[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)的条件：

①规格说明以[判定表](https://baike.baidu.com/item/%E5%88%A4%E5%AE%9A%E8%A1%A8)形式给出，或很容易转换成判定表。

②条件的排列顺序不会也不影响执行哪些操作。

③规则的排列顺序不会也不影响执行哪些操作。

④每当某一规则的条件已经满足，并确定要执行的操作后，不必检验别的规则。

⑤如果某一规则得到满足要执行多个操作，这些操作的执行顺序无关紧要。

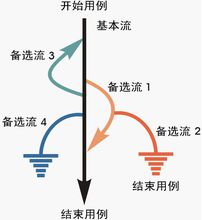
随机数法  
    即测试用例的参数是随机数。它可以自动生成，因此自动化程度高。使用大量随机测试用例测试通过的程序会提高用户对程序的信心。但其关键在于随机数的规律是否符合使用实际。

### 正交试验设计

就是使用已经造好了的正交[表格](https://baike.baidu.com/item/%E8%A1%A8%E6%A0%BC)来安排试验并进行数据分析的一种方法，目的是用最少的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)达到最高的测试覆盖率。

### 场景法

软件几乎都是用事件触发来控制流程的，事件触发的情景

[](https://baike.baidu.com/pic/%E9%BB%91%E7%9B%92%E6%B5%8B%E8%AF%95/934030/0/91ef76c6a7efce1b3b0b1a23af51f3deb48f6533?fr=lemma&ct=single)基本流和备选流

便形成了场景，而同一事件不同的触发顺序和处理结果就形成事件流。这种在软件设计方面的思想也可以引入到软件测试中，可以比较生动地描绘出事件触发时的情景，有利于测试设计者设计测试用例，同时使测试用例更容易理解和执行。

基本流和备选流：如下图所示，图中经过用例的每条路径都用基本流和备选流来表示，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（如备选流1和3）；也可能起源于另一个备选流（如备选流2），或者终止用例而不再重新加入到某个流（如备选流2和4）。[2]

## 流程

### 测试计划

首先，根据用户需求报告中关于功能要求和性能指标的规格说明书，定义相应的测试需求报告，即制订黑盒测试的最高标准，以后所有的测试工作都将围绕着测试需求来进行，符合测试需求的[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)即是合格的，反之即是不合格的；同时，还要适当选择测试内容，合理安排测试人员、测试时间及测试资源等。

### 测试设计

将[测试计划](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%AE%A1%E5%88%92)阶段制订的测试需求分解、细化为若干个可执行的[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)，并为每个测试过程选择适当的[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)（测试用例选择的好坏将直接影响到测试结果的有效性）。

### 测试开发

建立可重复使用的自动[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)。

### 测试执行

执行测试开发阶段建立的自动[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)，并对所发现的[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)进行[跟踪管理](https://baike.baidu.com/item/%E8%B7%9F%E8%B8%AA%E7%AE%A1%E7%90%86)。测试执行一般由[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95)、组合测试、[集成测试](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E6%B5%8B%E8%AF%95)、系统联调及[回归测试](https://baike.baidu.com/item/%E5%9B%9E%E5%BD%92%E6%B5%8B%E8%AF%95)等步骤组成，测试人员应本着科学负责的态度，[一步一个脚印](https://baike.baidu.com/item/%E4%B8%80%E6%AD%A5%E4%B8%80%E4%B8%AA%E8%84%9A%E5%8D%B0)地进行测试。

### 测试评估

结合量化的[测试覆盖](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%A6%86%E7%9B%96)域及[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)跟踪报告，对于[应用软件](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E8%BD%AF%E4%BB%B6)的质量和开发团队的工作进度及工作效率进行综合评价。

## 优点

1. 基本上不用人管着，如果[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)停止运行了一般就是被[测试程序](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%A8%8B%E5%BA%8F)crash了

2. 设计完[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)之后，下来的工作就是爽了，当然更苦闷的是确定crash原因

## 缺点

1. 结果取决于[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)的设计，测试用例的设计部分优势来源于经验，OUSPG的东西很值得借鉴

2. 没有状态转换的概念，一些成功的例子基本上都是针对PDU来做的，还做不到针对被[测试程序](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%A8%8B%E5%BA%8F)的状态转换来实现

3. 就没有状态概念的测试来说，寻找和确定造成[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)crash的测试例是个麻烦事情，必须把周围可能的测试例单独确认一遍。而就有状态的测试来说，就更麻烦了，尤其不是一个单独的testcase造成的问题。这些在堆的问题中表现的更为突出。

## 工具选择

如何高效地完成功能测试？选择一款合适的功能测试工具并培训一支高素质的[工具使用](https://baike.baidu.com/item/%E5%B7%A5%E5%85%B7%E4%BD%BF%E7%94%A8)队伍无疑是至关重要的。尽管现阶段存在少数不采用任何功能测试工具，从事功能测试外包项目的软件服务企业。短期来看，这类企业盈利状况尚可，但长久来看，它们极有可能被自动化程度较高的软件服务企业取代。

用于功能测试的[工具软件](https://baike.baidu.com/item/%E5%B7%A5%E5%85%B7%E8%BD%AF%E4%BB%B6)有很多，针对不同架构软件的工具也不断推陈出新。这里重点介绍的是其中一个较为典型[自动化测试](https://baike.baidu.com/item/%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95)工具，即Mercury公司的WinRunner。

WinRunner是一种用于检验[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)能否如期运行的企业级软件功能测试工具。通过自动捕获、检测和模拟用户交互操作，WinRunner能识别出绝大多数软件功能[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)，从而确保那些跨越了多个功能点和[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93)的[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)在发布时尽量不出现功能性故障。

WinRunner的特点在于： 与传统的[手工测试](https://baike.baidu.com/item/%E6%89%8B%E5%B7%A5%E6%B5%8B%E8%AF%95)相比，它能快速、批量地完成功能点测试； 能针对相同[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)，执行相同的动作，从而消除人工测试所带来的理解上的误差； 此外，它还能重复执行相同动作，测试工作中最枯燥的部分可交由机器完成； 它支持[程序](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F)风格的测试脚本，一个高素质的[测试工程师](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E5%B7%A5%E7%A8%8B%E5%B8%88)能借助它完成流程极为复杂的测试，通过使用[通配符](https://baike.baidu.com/item/%E9%80%9A%E9%85%8D%E7%AC%A6)、宏、条件语句、[循环语句](https://baike.baidu.com/item/%E5%BE%AA%E7%8E%AF%E8%AF%AD%E5%8F%A5)等，还能较好地完成测试脚本的重用； 它针对于大多数[编程语言](https://baike.baidu.com/item/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80)和Windows技术，提供了较好的集成、支持环境，这对基于Windows平台的[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)实施功能测试而言带来了极大的便利。

## 工作流程

### 识别GUI

在WinRunner中，我们可以使用GUI Spy来识别各种GUI[对象](https://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1)，识别后，WinRunner会将其存储到GUI Map File中。它提供两种GUI Map File模式： Global GUI Map File和GUI Map File per Test。其最大区别是后者对每个[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)产生一个GUI文件，它能自动建立、[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8)、加载，推荐初学者选用这种模式。但是，这种模式不易于描述对象的改变，其效率比较低，因此对于一个有经验的测试人员来说前者不失为一种更好的选择，它只产生一个共享的GUI文件，这使得[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)更容易维护，且效率更高。

### 建立测试脚本

在建立测试[脚本](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC)时，一般先进行录制，然后在录制形成的脚本中手工加入需要的TSL（与C语言类似的测试[脚本语言](https://baike.baidu.com/item/%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80)）。录制脚本有两种模式： Context Sensitive和Analog，选择依据主要在于是否对鼠标轨迹进行模拟，在需要回放时一般选用Analog。在录制过程中这两种模式可以通过F2键相互切换。

只要看看现代软件的规模和功能点数就可以明白，功能测试早已跨越了单靠手工敲敲键盘、点点鼠标就可以完成的阶段。而[性能测试](https://baike.baidu.com/item/%E6%80%A7%E8%83%BD%E6%B5%8B%E8%AF%95)则是控制系统性能的有效手段，在软件的能力验证、能力规划、[性能调优](https://baike.baidu.com/item/%E6%80%A7%E8%83%BD%E8%B0%83%E4%BC%98)、[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)修复等方面都发挥着重要作用。

### 脚本除错

在WinRunner中有专门一个Debug Toolbar用于[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)除错。可以使用step、pause、breakpoint等来控制和跟踪[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)和查看各种[变量](https://baike.baidu.com/item/%E5%8F%98%E9%87%8F)值。

### 测试脚本

当[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)有新版本发布时，我们会对应用程序的各种功能包括新增功能进行测试，这时当然不可能再来重新录制和编写所有的[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)。我们可以使用已有的脚本，批量运行这些[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)测试旧的功能点是否正常工作。可以使用一个call命令来加载各[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)。还可在call命令中加各种TSL脚本来增加批量能力。

### 分析测试结果

分析测试结果在整个[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)中最重要，通过分析可以发现[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)的各种功能性[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)。当运行完某个[测试脚本](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%84%9A%E6%9C%AC)后，会产生一个[测试报告](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E6%8A%A5%E5%91%8A)，从这个测试报告中我们能发现[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)的功能性[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)，能看到实际结果和期望结果之间的差异，以及在[测试过程](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E8%BF%87%E7%A8%8B)中产生的各类对话框等。

### 回报缺陷

在分析完[测试报告](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E6%8A%A5%E5%91%8A)后，按照测试流程要回报[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)的各种[缺陷](https://baike.baidu.com/item/%E7%BC%BA%E9%99%B7)，然后将这些缺陷发给指定人，以便进行修改和维护。

## 常用方法

功能测试就是对产品的各功能进行验证，根据功能[测试用例](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E7%94%A8%E4%BE%8B)，逐项测试，检查产品是否达到用户要求的功能。常用的测试方法如下

1. 页面链接检查：每一个链接是否都有对应的页面，并且页面之间切换正确。

2. 相关性检查：删除/增加一项会不会对其他项产生影响，如果产生影响，这些影响是否都正确。

3. 检查按钮的功能是否正确：如update,cancel,delete,save等功能是否正确。

4. 字符串长度检查： 输入超出需求所说明的字符串长度的内容，看系统是否检查字符串长度，会不会出错.

5. 字符类型检查： 在应该输入指定类型的内容的地方输入其他类型的内容（如在应该输入整型的地方输入其他字符类型），看系统是否检查字符类型，会否报错.

6. 标点符号检查： 输入内容包括各种标点符号，特别是空格，各种引号，[回车键](https://baike.baidu.com/item/%E5%9B%9E%E8%BD%A6%E9%94%AE).看系统处理是否正确.

7. 中文[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)处理： 在可以输入中文的系统输入中文，看会否出现[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)或出错.

8. 检查带出信息的完整性： 在查看信息和update信息时，查看所填写的信息是不是全部带出.，带出信息和添加的是否一致

9. 信息重复： 在一些需要命名，且名字应该唯一的信息输入重复的名字或ID，看系统有没有处理，会否报错，重名包括是否区分大小写，以及在输入内容的前后输入空格，系统是否作出正确处理.

10. 检查删除功能：在一些可以一次删除多个信息的地方，不选择任何信息，按”delete”，看系统如何处理，会否出错；然后选择一个和多个信息，进行删除，看是否正确处理.

11. 检查添加和修改是否一致： 检查添加和修改信息的要求是否一致，例如添加要求必填的项，修改也应该必填；添加规定为整型的项，修改也必须为整型.

12. 检查修改重名：修改时把不能重名的项改为已存在的内容，看会否处理，报错.同时，也要注意，会不会报和自己重名的错.

13. 重复提交[表单](https://baike.baidu.com/item/%E8%A1%A8%E5%8D%95)：一条已经成功提交的纪录，back后再提交，看看系统是否做了处理。

14. 检查多次使用back键的情况： 在有back的地方，back，回到原来页面,再back，重复多次，看会否出错.

15. search检查： 在有search功能的地方输入系统存在和不存在的内容，看search结果是否正确.如果可以输入多个search条件，可以同时添加合理和不合理的条件，看系统处理是否正确.

16. 输入信息位置： 注意在[光标](https://baike.baidu.com/item/%E5%85%89%E6%A0%87)停留的地方输入信息时，光标和所输入的信息会否跳到别的地方.

17. 上传下载文件检查：上传下载文件的功能是否实现，上传文件是否能打开。对上传文件的格式有何规定，系统是否有解释信息，并检查系统是否能够做到。

18. 必填项检查：应该填写的项没有填写时系统是否都做了处理，对必填项是否有提示信息，如在必填项前加\*

19. [快捷键](https://baike.baidu.com/item/%E5%BF%AB%E6%8D%B7%E9%94%AE)检查：是否支持常用快捷键，如Ctrl+C Ctrl+V Backspace等，对一些不允许输入信息的字段，如选人，选日期对[快捷方式](https://baike.baidu.com/item/%E5%BF%AB%E6%8D%B7%E6%96%B9%E5%BC%8F)是否也做了限制。

20. [回车键](https://baike.baidu.com/item/%E5%9B%9E%E8%BD%A6%E9%94%AE)检查： 在输入结束后直接按回车键，看系统处理如何，会否报错。

其他参考：<https://blog.csdn.net/zhang_xiaoyi/article/details/79017460>

<https://blog.csdn.net/qq_36626686/article/details/79705585>

<http://www.cnblogs.com/python-xiakaibi/p/8445864.html>

参考资料

* **1.****万年红 李翔 ．软件黑盒测试的方法与实践： 《计算机工程》，2000年12期**
* **2.****请参阅《史上最全测试用例方法总结》**