黑盒测试的测试用例设计方法/软件测试的 14 种类型

一、黑盒测试的测试用例设计方法

等价类划分

是把所有可能的输入数据,即程序的输入域划分成若干部分（子集）,然后从每一个子集中选取少数具有代表性的数据作为测试用例.该方法是一种重要的,常用的黑盒测试用例设计方法.

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: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.

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.

③在输入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.

④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.

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（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: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 有效等价类 无效等价类

… … …

… … …

然后从划分出的等价类中按以下三个原则设计测试用例:

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,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.

边界值分析法

边界值分析方法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,可以查出更多的错误.

使用边界值分析方法设计测试用例,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.

6）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的边界上的值作为测试用例.

7）分析规格说明,找出其它可能的边界条件.

错误推测法

基于经验和直觉推测程序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的方法.

错误推测方法的基本思想: 列举出程序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. 例如, 在单元测试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.

因果图方法

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. 这就需要利用因果图（逻辑模型）.

因果图方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.

利用因果图生成测试用例的基本步骤:

(1) 分析软件规格说明描述中, 那些是原因(即输入条件或输入条件的等价类),那些是结果(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.

(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不可能出现. 为表明这些特殊情况, 在因果图上用一些记号表明约束或限制条件.

(4) 把因果图转换为判定表.

(5) 把判定表的每一列拿出来作为依据,设计测试用例.

从因果图生成的测试用例（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.

前面因果图方法中已经用到了判定表.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.在程序设计发展的初期,判定表就已被当作编写程序的辅助工具了.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.

判定表通常由四个部分组成.

条件桩（Condition Stub）:列出了问题得所有条件.通常认为列出得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:（根据软件规格说明）

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 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设计测试用例的条件:

①规格说明以判定表形式给出,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.

二、软件测试的 14 种类型

软件测试是指使用人工或者自动的手段来运行或测定某个软件产品系统的过程，其目的是在于检验是否满足规定的需求或者弄清预期的结果与实际结果的区别。本文主要描述软件测试的类型。

1 数据和数据库完整性测试

数据与数据库完整测试是指测试关系型数据库完整性原则以及数据合理性测试。

数据库完整性原即：

主码完整性：主码不能为空；

外码完整性：外码必须等于对应的主码或者为空。

数据合理性指数据在数据库中的类型，长度，索引等是否建的比较合理。

在项目名称中，数据库和数据库进程应作为一个子系统来进行测试。在测试这些子系统时，不应将测试对象的用户界面用作数据的接口。对于数据库管理系统 (DBMS)，还需要进行深入的研究，以确定可以支 1 持测试的工具和技术。

比如，有两张表：部门和员工。部门中有部门编号，部门名称，部门经理等字段，主码为部门编号；员工表中有员工编号，员工所属部门编号，员工名称，员工类型等字段，主码为员工编号，外码为员工所属部门编号，对应部门表。如果在某条部门记录中部门编号或员工记录员工编号为空，他就违反主码完整性原则。如果某个员工所属部门的编号为##，但是##在部门编号中确找不到，这就违反外码完整性原则。

员工类型如下定义：0：职工，1：职员，2：实习生。但数据类型为 Int，我们都知道 Int 占有 4 个字节，如果定义成 char(1).就比原来节约空间。

2 白盒测试

白盒测试是基于代码的测试，测试人员通过阅读程序代码或者通过使用开发工具中的单步调试来

判断软件的质量，一般黑盒测试由项目经理在程序员开发中来实现。白盒测试分为动态白盒测试和静态白盒测试

2.1 静态白盒测试

利用眼睛，浏览代码，凭借经验，找出代码中的错误或者代码中不符合书写规范的地方。比如，代码规范中规定，函数必须为动宾结构。而黑盒测试发现一个函数定义如下：

```
Function NameGet(){
….
}
```

这是属于不符合开发规范的错误。

有这样一段代码:

```
if (i<0) & (i>=0)
…
```

这段代码交集为整个数轴，IF 语句没有必要

```
I=0;
while(I>100){
J=J+100;
T=J*PI;
}
```

在循环体内没有 I 的增加,bug 产生。

2.2 动态白盒测试

利用开发工具中的调式工具进行测试。比如一段代码有 4 个分支，输入 4 组不同的测试数据使 4 组分支都可以走通而且结果必须正确。

看一段代码

```
if(I<0){
P1
}else{
P2
```

}

在调试中输入 I=-1,P1 程序段通过， P2 程序段未通过，属于动态黑盒测试的缺陷

3.功能测试

功能测试指测试软件各个功能模块是否正确，逻辑是否正确。

对测试对象的功能测试应侧重于所有可直接追踪到用例或业务功能和业务规则的测试需求。这种测试的目标是核实数据的接受、处理和检索是否正确，以及业务规则的实施是否恰当。此类测试基于黑盒技术，该技术通过图形用户界面 (GUI) 与应用程序进行交互，并对交互的输出或结果进行分析，以此来核实应用程序及其内部进程。功能测试的主要参考为类似于功能说明书之类的文档。

比如一个对电子商务系统，前台用户浏览商品-放入购物车-进入结账台，后台处理订单，配货，付款，发货，这一系列流程必须正确无误的走通，不能存在任何的错误。

4.UI 测试

UI 测试指测试用户界面的风格是否满足客户要求，文字是否正确，页面美工是否好看，文字，图片组合是否完美，背景是否美观，操作是否友好等等
用户界面 (UI) 测试用于核实用户与软件之间的交互。UI 测试的目标是确保用户界面会通过测试对象的功能来为用户提供相应的访问或浏览功能。另外，UI 测试还可确保 UI 中的对象按照预期的方式运行，并符合公司或行业的标准。包括用户友好性，人性化，易操作性测试。UI 测试比较主观，与测试人员的喜好有关

比如：页面基调颜色刺眼；用户登入页面比较难于找到，文字中出现错别字，页面图片范围太广等都属于 UI 测试中的缺陷，但是这些缺陷都不太严重。

2 软件测试的 14 种类型

5.性能测试

性能测试主要测试软件测试的性能，包括负载测试，强度测试，数据库容量测试，基准测试以及

基准测试

5.1 负载测试

负载测试是一种性能测试指数据在超负荷环境中运行，程序是否能够承担。

在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下仍能正常运行。此外，负载测试还要评估性能特征，例如，响应时间、事务处理速率和其他与时间相关的方面。

比如，在 B/S 结构中用户并发量测试就是属于负载测试的用户，可以使用 webload 工具，模拟上百人客户同时访问网站，看系统响应时间，处理速度如何？

5.2 强度测试

强度测试是一种性能测试，他在系统资源特别低的情况下软件系统运行情况。这类测试往往可以书写系统要求的软硬件水平要求。

实施和执行此类测试的目的是找出因资源不足或资源争用而导致的错误。如果内存或磁盘空间不足，测试对象就可能会表现出一些在正常条件下并不明显的缺陷。而其他缺陷则可能由于争用共享资源（如数据库锁或网络带宽）而造成的。强度测试还可用于确定测试对象能够处理的最大工作量。

比如：一个系统在内存 366M 下可以正常运行，但是降低到 258M 下不可以运行，告诉内存不足，这个系统对内存的要求就是 366M。

5.3 数据库容量测试

数据库容量测试指通过存储过程往数据库表中插入一定数量的数据，看看相关页面是否能够及时显示数据。

数据库容量测试使测试对象处理大量的数据，以确定是否达到了将使软件发生故障的极限。容量测试还将确定测试对象在给定时间内能够持续处理的最大负载或工作量。例如，如果测试对象正在为

生成一份报表而处理一组数据库记录，那么容量测试就会使用一个大型的测试数据库，检验该软件是否正常运行并生成了正确的报表。做这种测试通常通过书写存储过程向数据库某个表中插入一定数量的记录，计算相关页面的调用时间。

比如，在电子商务系统中，通过 insert customer 往 user 表中插入 10 000 数据，看其是否可以正常显示顾客信息列表页面，如果要求达到最多可以处理 100 000 个客户，但是顾客信息列表页面不能够在规定的时间内显示出来，就需要调整程序中的 SQL 查询语句；如果在规定的时间内显示出来，可以将用户数分别提高到 20 000 , 50 000, 100 000 进行测试。

5.4 基准测试

基准测试与已知现有的系统进行比较，主要检验是否与类似的产品具有竞争性的一种测试。

如果你要开发一套财务系统软件并且你已经获得用友财务系统的性能等数据，你可以测试你这套系统，看看哪些地方比用友财务系统好，哪些地方差？以便改进自己的系统，也可为产品广告提供数据。

5.5 竞争测试

软件竞争使用各种资源（数据纪录，内存等），看他与其他相关系统对资源的争夺能力。比如：一台机器上即安装您的财务系统，又安装用友财务系统。当 CPU 占有率下降后，看看是否能够强过用友财务系统，而是自己的系统能够正常运行？

6. 安全性和访问控制测试

安全性和访问控制测试侧重于安全性的两个关键方面：

应用程序级别的安全性，包括对数据或业务功能的访问

系统级别的安全性，包括对系统的登录或远程访问。

6.1 应用程序级别的安全性

可确保：在预期的安全性情况下，主角只能访问特定的功能或用例，或者只能访问有限的数据。

例如，可能会允许所有人输入数据，创建新账户，但只有管理员才能删除这些数据或账户。如果具有数据级别的安全性，测试就可确保“用户类型一”能够看到所有客户消息（包括财务数据），而“用户二”只能看见同一客户的统计数据。

比如 B/S 系统，不通过登入页面，直接输入 URL,看其是否能够进入系统？

6.2 系统级别的安全性

可确保只有具备系统访问权限的用户才能访问应用程序，而且只能通过相应的网关来访问。

3 软件测试的 14 种类型

比如输入管理员账户，检查其密码是否容易猜取，或者可以从数据库中获得？

7.故障转移和恢复测试

故障转移和恢复测试指当主机软硬件发生灾难时候，备份机器是否能够正常启动，使系统是否可以正常运行，这对于电信，银行等领域的软件是十分重要的。

故障转移和恢复测试可确保测试对象能成功完成故障转移，并能从导致意外数据损失或数据完整性破坏的各种硬件、软件或网络故障中恢复。

故障转移测试可确保：对于必须持续运行的系统，一旦发生故障，备用系统就将不失时机地“顶替”发生故障的系统，以避免丢失任何数据或事务。

恢复测试是一种对抗性的测试过程。在这种测试中，将把应用程序或系统置于极端的条件下（或者是模拟的极端条件下），以产生故障（例如设备输入/输出 (I/O) 故障或无效的数据库指针和关健字）。然后调用恢复进程并监测和检查应用程序和系统，核实应用程序或系统和数据已得到了正确的恢复。

一定要注意主备定时备份

比如电信系统，突然主机程序发生死机，备份机器是否能够启动，使系统能够正常运行，从而不

影响用户打电话？

8.配置测试

又叫兼容性测试。配置测试核实测试对象在不同的软件和硬件配置中的运行情况。在大多数生产环境中，客户机工作站、网络连接和数据库服务器的具体硬件规格会有所不同。客户机工作站可能会安装不同的软件例如，应用程序、驱动程序等而且在任何时候，都可能运行许多不同的软件组合，从而占用不同的资源。（如浏览器版本，操作系统版本等）

下面列出主要配置测试

8.1 浏览器兼容性

测试软件在不同产商的浏览器下是否能够正确显示与运行；

比如测试 IE，Natscape 浏览器下是否可以运行这套软件？

8.2 操作系统兼容性

测试软件在不同操作系统下是否能够正确显示与运行；

比如测试 WINDOWS98,WINDOWS 2000,WINDOWS XP,LINU, UNIX 下是否可以运行这套软件？

8.3 硬件兼容性

测试与硬件密切相关的软件产品与其他硬件产品的兼容性，比如该软件是少在并口设备中的，测试同时使用其他并口设备，系统是否可以正确使用.
比如在 INTER,舒龙 CPU 芯片下系统是否能够正常运行？

这样的测试必须建立测试实验室，在各种环境下进行测试。

9.安装测试

安装测试有两个目的。第一个目的是确保该软件在正常情况和异常情况的不同条件下: 例如，进行首次安装、升级、完整的或自定义的安装_都能进行安装。异常情况包括磁盘空间不足、缺少目录创

建权限等。第二个目的是核实软件在安装后可立即正常运行。这通常是指运行大量为功能测试制定的测试。
安装测试包括测试安装代码以及安装手册。安装手册提供如何进行安装，安装代码提供安装一些程序能够运行的基础数据。

10.多语种测试

又称本地化测试，是指为各个地方开发产品的测试，如英文版，中文版等等，包括程序是否能够正常运行，界面是否符合当地习俗，快捷键是否正常起作用等等，特别测试在 A 语言环境下运行 B 语言软件（比如在英文 win98 下试图运行中文版的程序），出现现象是否正常。

本地化测试还要考虑：

- 当语言从 A 翻译到 B，字符长度变化是否影响页面效果。比如中文软件中有个按键叫“看广告”，翻译到英文版本中为 “View advertisement”可能影响页面的美观程度
- 要考虑同一单词在各个国家的不同意思，比如 football 在英文中为足球，而美国人使用中可能理解为美式橄榄球。
- 要考虑各个国家的民族习惯，比如龙个美国中被理解邪恶的象征，但翻译到中国，中国人认为为吉祥的象征。

11.文字测试

文字测试测试软件中是否拼写正确，是否易懂，不存在二义性，没有语法错误；文字与内容是否有出入等等，包括图片文字。
比如：“比如，请输入正确的证件号码！”何谓正确的证件号码，证件可以为身份证，驾驶证，也可为军官证，如果改为“请输入正确的身份证号码！”用户就比较容易理解了。

12.分辨率测试

测试在不同分辨率下，界面的美观程度,分为 800*600，1024*768，1152*864，1280*768，1280*1024，1200*1600 大小字体下测试。一个好的软件要有一个极佳的分辨率，而在其他分辨率下也都能可以运行。

13 发布测试

主要在产品发布前对一些附带产品，比如说明书，广告稿等进行测试

13.1 说明书测试

主要为语言检查，功能检查，图片检查

语言检查：检查说明书语言是否正确，用词是否易于理解；

功能检查：功能是否描述完全，或者描述了并没有的功能等；

图片检查:：检查图片是否正确

13.2 宣传材料测试

主要测试产品中的附带的宣传材料中的语言，描述功能，图片

13.3 帮助文件测试

帮助文件是否正确，易懂，是否人性化。最好能够提供检索功能。

13.4 广告用语

产品出公司前的广告材料文字，功能，图片，人性化的检查

14 文档审核测试

文档审核测试目前越来越引起人们的重视，软件质量不是检查出来的，而是融进软件开发中来。前置软件测试发越来越受到重视。请看一个资料：

文档审核测试主要包括需求文档测试，设计文档测试，为前置软件测试测试中的一部分。

14.1 需求文档测试

主要测试需求中是否存在逻辑矛盾以及需求在技术上是否可以实现；

14.2 设计文档测试

测试设计是否符合全部需求以及设计是否合理。

总结

据美国软件质量安全中心 2000 年对美国一百家知名的软件厂商统计，得出这样一个结论：软件缺陷在开发前期发现比在开发后期发现资金，人力上节约 90%；软件缺陷在推向市场前发现比在推出后发现资金，人力上节约 90%。所以说软件的缺陷应该尽早发现。不是所有的软件都要进行任何类型的软件测试的，可以根据产品的具体情况进行组装测试不同的类型。