DOI:10.16184/j.cnki.comprg.2010.04.010



# 软件测试用例设计

李洁

(四川司法警官职业学院，德阳 618000)

摘 要：软件测试重要性越来越大，测试用例的设计是整个测试过程的基础。结合工作实践介绍软件测试用例的重要性和测试用例的设计方法，并举例说明如何应用白盒测试技术和黑盒测试技术。

关键词：软件测试；白盒测试；黑盒测试；测试用例

# Design the Software Test Case

LI Jie

(Sichuan Judicial and Police Officers´ Professional College, Deyang 618000)

**Abstract:** Testing becomes more and more important for software. To design a test case is the very beginning of the testing procedure. In this essay, how significant the software test case is and how to design the cases are introduced, with working practice combined, and moreover, how to apply white box testing and black box testing is illustrated here.

**Key words:** software testing; white box testing; black box testing; test cases

## 1 引言

随着计算机与人们日常生活关系日益紧密，各种类型的软件被广泛应用于各行各业之中。人们对软件的依赖性越大，就会对软件的品质要求越来越高。在这样的背景下，近年来，在国内，软件测试越来越受到软件领域的关注。

软件测试是软件生命周期中的一个重要阶段，它是软件品质得以保证的重要过程，是根据软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例，并利用这些测试用例运行软件，以发现软件错误的过程。它是帮助识别开发完成的计算机软件的正确度、完全度和质量的软件过程；是软件质量保证 (SQA) 的重要子域。

软件测试用例就是在软件测试过程中为特定的目的而设计的一组测试输入、执行条件和预期的结果。测试用例是执行的最小实体。简单地说，测试用例就是设计一个场景，使软件程序在这种场景下，必须能够正常运行并且达到程序所设计的执行结果。

## 2 软件测试用例的重要性

如何保障软件测试质量的稳定？有了测试用例，无论是谁来测试，参照测试用例实施，都能保障测试的质量，这样就可以把人为因素的影响减少到最小。即便最初的测试用例考虑不周全，随着测试的进行和软件版本更新，也将日趋完善。

因此在测试工作中，测试用例的设计是非常重要的，是测试执行的正确性、有效性的基础。如何有效地设计测试用例，一直是测试人员所关注的问题；设计好测试用例，也是保证测试工作的关键因素之一。

## 3 测试用例设计

### 3.1 文档

在编写测试用例文档时应当有文档模板，且文档模板须符合内部的规范要求，测试用例文档将受制于测试用例管理软件的约束。软件产品或软件开发项目的测试用例一般以该产品的软件模块或子系统为单位，形成一个测试用例文档，但并不是绝对的。

测试用例文档由简介和测试用例两部分组成。简介部分编制测试目的、测试范围、定义术语、参考文档、概述等；测试用例部分逐一列示各测试用例。每个具体测试用例都将包括下列详细信息：用例编号、用例名称、测试等级、入口准则、验证步骤、期望结果 (含判断标准)、出口准则、注释等。以上内容涵盖了测试用例的基本元素：测试索引、测试环境、测试输入、测试操作、预期结果、评价标准。

### 3.2 设置

早期的测试用例是按功能设置用例。后来引进了路径分析法，按路径设置用例。目前演变为按功能、路径混合模式设置用例。

按功能测试是最简捷的，按用例规约遍历测试每一功能。对于复杂操作的程序模块，其各功能的实施是相互影响、紧密相关、环环相扣的，可以演变出数量繁多的变化。没有严密的逻辑分析，产生遗漏是在所难免。路径分析是一个很好的方法，其最大的优点是在于可以避免漏测试。但路径分析法也有局限性，在一个非常简单字典维护模块中就存在 10 余条路径，一个复杂的模块中会有几十到上百条路径是不足为奇的。

### 3.3 设计

测试用例可以分为基本事件、备选事件和异常事件。设计基本事件的用例，应该参照用例规约 (或设计规格说明书)，根据关联的功能、操作按路径分析法设计测试用例。而对孤立的功能则直接按功能设计测试用例。基本事件的测试用例

收稿日期：2009-11-16

应包含所有需要实现的需求功能，覆盖率达 100%。

3.4 一般原则

通常，在设计测试用例时应遵循以下原则：

⑴ 应该避免依赖先前测试用例的输出，因为测试用例的执行序列早期发现的错误可能导致其他的错误，从而减少测试执行时实际测试的代码量。

⑵ 测试用例设计过程中，包括作为试验执行这些测试用例时，常常可以在软件构建前就发现 BUG。还有可能在测试设计阶段比测试执行阶段发现更多的 BUG。

⑶ 在整个单元测试设计中，主要的输入应该是被测单元的设计文档。在某些情况下，需要将试验实际代码作为测试设计过程的输入，测试设计者必须意识到不是在测试代码本身。从代码构建出来的测试说明只能证明代码执行完成的工作，而不是代码应该完成的工作。

## 4 测试用例设计方法

设计测试用例，也分为黑盒设计方法和白盒设计方法。黑盒设计方法分为等价类划分法、边界值划分法、错误推测法、因果图法等，而白盒设计方法又分为逻辑覆盖法和基本路径覆盖法，或者分为语句覆盖、判定覆盖、条件覆盖方法。在实际测试用例设计过程中，不仅根据需要、场合单独使用这些方法，常常综合运用多个方法，使测试用例的设计更为有效。测试用例设计最重要的因素是经验和常识。测试设计者不应该让某种测试技术阻碍经验和常识的运用。

4.1 黑盒测试方法及测试用例设计

黑盒测试用例设计：使用详细设计导出测试用例。

采用黑盒测试的目的主要是：检查功能是否实现或遗漏；检查人机界面是否错误；数据结构或外部数据库访问错误；性能等其它特性要求是否满足；初始化和终止错误。

黑盒测试用例设计时应该注意以下问题：

⑴ 测试用例要根据测试大纲来编写。

⑵ 测试用例也要分测试项进行归类，这样比较好分析和阅读。如：业务流程测试、安装测试、功能测试、用户友好性测试、兼容性测试、性能测试、安全性测试等等。

⑶ 编写测试用例时要考虑各种情况，精力应主要集中在软件的主要业务流程和风险高的地方。

⑷ 熟悉系统，对编写测试用例很有帮助。

黑盒测试法侧重于对被测软件功能的检测，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适当的输出信息，并且保持外部信息的完整性。黑盒测试法主要着眼于程序外部结构，不考虑程序内部逻辑结构，主要针对软件界面、软件功能、外部数据库访问及软件初始化等方面进行测试。具体地说黑盒测试方法主要包括边界值分析法、等价类划分法、比较测试法、因果图法、错误推错法、判定表驱动法等。

下面将对边界值分析法和等价类划分方法做一些探讨。

4.1.1 边界值分析法

边界值分析法是一种经常使用的软件测试技术，它具有很强的发现程序错误的能力。为检测边界附近的处理专门设计测试用例，通常都会取得很好的测试效果。

在用这种方法进行测试时要注意以下一些原则：

⑴ 如果输入条件对取值范围进行了界定，则应该以边界内部以及恰巧超出范围边界外的值作为测试用例。

⑵） 如果随取值的个数进行了界定，则应当分别以最大、最小个数及稍小于最小、稍大于最大个数作为测试用例。

⑶ 对于输出值同样可以按照上面两条原则来设计测试用例。

⑷ 若在程序规格说明书中提到的输入输出域是一个有序的集合，就应该选取该有序集合中的第一个和最后一个元素作为测试用例。

测试用例的设计只要按照以上原则设计就可以取得较好的效果。

4.1.2 等价类划分方法

等价类划分方法是一种典型的黑盒测试方法，它完全不考虑程序的内部结构，而只是根据对程序的说明和要求来进行测试用例的设计。测试人员先要对需求规格说明书的各项需求，尤其是功能需求进行细致分析，同时，还要求对输入要求和输出要求区别对待和处理。

等价类划分方法把程序的输入域划分成若干部分，然后从每个部分中选取出少数代表性数据当作测试用例。这主要是因为完成穷举测试实际上是不可能完成的工作，因此，测试人员要从大量的可能数据中选取其中一部分数据，用来作为测试用例。经过划分后，每一类的代表性数据在测试中的作用都等价于这一类中的其他值。也就是说，如果某一类中的一个测试用例发现了错误，那么这一等价类中的其他用例也能发现同样的错误；与之相反的是，如果某一类中的测试用例没有发现错误，则这一类中的其他测试用例也不会查出错误。在采用等价类划分方法来进行测试用例设计时，必须首先在分析需求规格说明的基础上划分等价类，列出等价类表，从而确定测试用例。

4.2 白盒测试方法及测试用例设计

白盒测试用例设计：使用程序设计的控制结构导出测试用例。

采用白盒测试的目的主要是保证一个模块中的所有独立路径至少被执行一次；对所有的逻辑值均需要测试真、假两个分支；在上下边界及可操作范围内运行所有循环；检查内部数据结构以确保其有效性。白盒测试用例设计的主要方法有基本路径测试、程序结构分析、逻辑覆盖等，它根据程序的控制结构设计测试用例，主要用于软件验证。

应用白盒测试方法需要遵循如下的几条原则：

⑴ 一个模块中的所有独立路径至少被测试一次。

⑵ 所有逻辑值均需测试 true 和 false 两种情况。

⑶ 检查程序的内部数据结构，保证其结构的有效性。

⑷ 在取值上下边界及可操作范围内运行所有循环。

任何有关路径分析的测试都可以称为路径测试。可以给出对路径测试的简单的描述，路径测试就是指从一个程序的入口开始，执行所经历的各个语句的完整过程。路径测试是白盒测试中最为典型的问题，完成路径测试的理想情况是做到路径覆盖。对于比较简单的小程序实现路径覆盖是可能做到的，但是程序中如果出现多个判定、多个循环，路径数目将会急剧地增长，不可能实现路径覆盖。

基本路径测试是一种白盒测试技术。其基本思想是测试用例设计者导出一个过程设计的逻辑复杂性测度，并使用该测度作为指南来定义执行路径的基本集，从该基本集导出的测试用例保证对程序中的每一条执行语句至少执行一次。

基本路径测试的方法步骤如下：

⑴ 画出控制流图

图 1 中的每一个圆称为流图的节点，代表一条或多条语句。流图中的箭头称为边或连接，代表控制流。任何过程设计都要被翻译成控制流图。如下面的 C 函数：

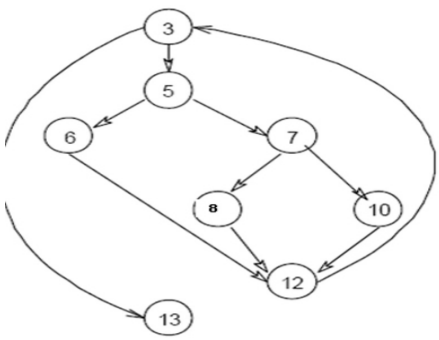


图 1 控制流图

```
void Sort (int iRecordNum,int iType)
{
1     int x=0;
2     int y=0;
3     while (iRecordNum--)
4     {
5         if (iType==0)
6             x=y+2;
7         else
8             if (iType==1)
9                 x=y+10;
10            else
11                x=y+20;
12    }
13}
```

值得注意的是，如果在程序中遇到复合条件，例如条件语句中的多个布尔运算符（逻辑 OR、AND）时，为每一个条件创建一个独立的节点，包含条件的节点称为判定节点，从每一个判定节点发出两条或多条边。例如：

```
if (a or b)
x
else
y
...
```

对应的逻辑如图 2 所示。

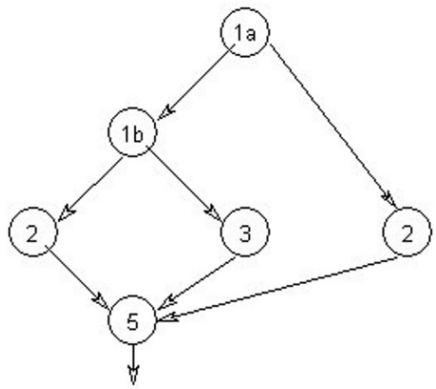


图 2 判定逻辑图

⑵ 计算圈复杂度

圈复杂度是一种为程序逻辑复杂性提供定量测度的软件度量，将该度量用于计算程序的基本的独立路径数目，为确保所有语句至少执行一次的测试数量的上界。独立路径必须包含一条在定义之前不曾用到的边。

有以下 3 种方法计算圈复杂度：

① 流图中区域的数量对应于环型的复杂性。

② 给定流图 G 的圈复杂度-V (G)，定义为 V (G) =E-N+2，E 是流图中边的数量，N 是流图中节点的数量。

③ 给定流图 G 的圈复杂度-V (G)，定义为 V (G) =P+1，P 是流图 G 中判定节点的数量。

对应上述图 1 中代码的圈复杂度，计算如下：

V (G) =10 条边-8 节点+2=4；

V (G) =3 个判定节点+1=4。

可以看出，控制流图中有四个区域；

⑶ 导出独立路径

根据上面的计算方法，可得出四个独立的路径：

路径 1：3-13

路径 2：3-5-6-12-3-13

路径 3：3-5-7-8-12-3-13

路径 4：3-5-7-10-12-3-13

⑷ 设计测试用例

根据上面的独立路径，设计输入数据，使程序分别执行到上面四条路径。如：

Sort (0,1) 测试路径 1

Sort (1,0) 测试路径 2

Sort (1,1) 测试路径 3

Sort (1,2) 测试路径 4

（下转到 23 页）

输出

```
                end case;
            end if;
        end if;
        end if;
    end process;
```

编译完成后对部分数据进行仿真，随机测试 3 个误码，错误位置分别在第一位（图 5），第五位（图 6）以及第七位（图 7），均能实现纠错译码；输入正确码元（图 8），亦能得到正确结果。

仿真结果如下：

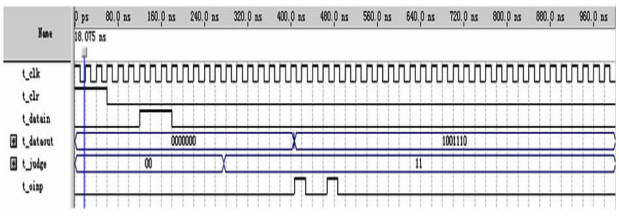

图 5 第一位错误 输入为 0001110,纠错后输出 1001

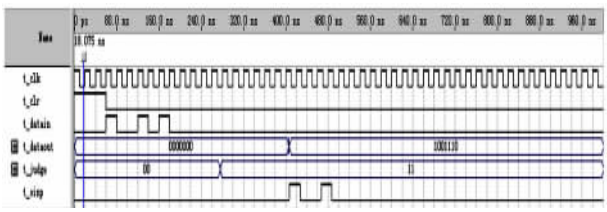

图 6 第五位错误 输入为 1001010,纠错后输出 1001

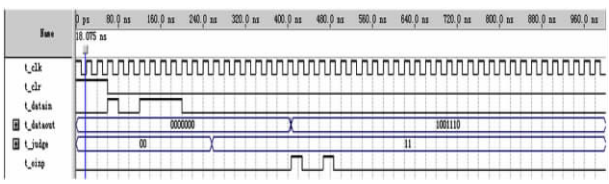

图 7 第七位错误 输入为 1001111,纠错后输出 1001

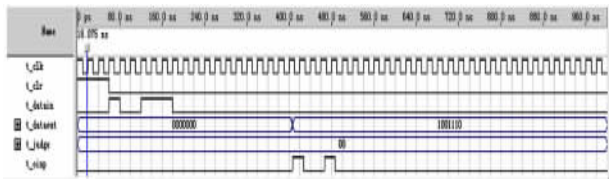

图 8 正确输入 输入为 1001110,输出 1001

## 4 结语

运用 VHDL 设计编译码器具有效率高、结果容易仿真、信号观察方便等特点，能把复杂的工程通过简练的、高层次的描述实现。此外，只要将编好的程序稍加改动，就可以实现更为复杂的电路，而不必从头做起，这样可以极大地提高工作效率。

## 参考文献

[1] 樊昌信，曹丽娜. 通信原理. 第 6 版，国防工业出版社，2008：340–346.

[2] 潘松，黄继业. EDA 技术与 VHDL. 第 2 版，清华大学出版社，2007.

[3] 佩德罗尼. VHDL 数字电路设计教程 [M] . 电子工业出版社，2005.

作者简介

刘嘉湘，男（1987-），主要研究方向：通信码元处理与 EDA 技术。

（上接第 19 页）

## 5 结语

针对软件测试过程中的黑盒测试和白盒测试的测试用例设计作了一些探讨，但影响软件测试的因素很多，例如软件本身的复杂程度、开发人员的素质、测试方法和技术的运用等。因为有些因素是客观存在的，有些因素则是波动的、不稳定的，工作状态也会受到情绪等因素的影响等，有了测试用例，就能在很大程度上保障测试的质量，提高软件的品质。

## 参考文献

[1] 郑人杰. 实用软件工程 [M] . 北京：清华大学出版社，2002.

[2] 王春森. 高级设计师教程 [M] . 北京：清华大学出版社，2001.

[3] 沃特金斯，贺红卫.实用软件测试过程 [M] .北京：机械工业出版社，2004.

[4] 柳纯露，黄子河. 软件评测师教程 [M] . 北京：清华大学出版社，2005.

[5] 韩万江，姜立新. 软件开发项目管理 [M] . 北京：机械工业出版社，2004.

[6] 王健. 软件测试员培训教材 [M] . 北京：电子工业出版社，2007.

作者简介

李洁，女（1975-），讲师，硕士，研究方向：计算机应用。