# 软件测试及测试用例的设计

罗 辉 (湖南省双峰工商银行)

软件测试是为尽可能多地发现软件的错误而进行的测试。它在软件生命周期中占据重要地位,这不仅是因为测试阶段所占用的时间、花费的人力和成本占软件开发的很大比重,而且它直接影响着软件的质量。如果在测试阶段未能很好地把握质量关,很可能对整个系统造成十分严重的后果。譬如美国一次卫星发射,控制程序中一个逗号",”误为句号"."没有测试出,导致火箭爆炸,直接经济损失达1000万美元! 软件测试在软件开发中的重要性由此可见一斑。

## 一、软件测试的原则

软件测试的成功与否,很大程度与测试者的心理有关,因此我们应坚持一定的测试原则。

1.程序测试是一个破坏性过程,其目的是为发现程序中的错误,但并不能保证软件不再存在错误。

2.程序员和程序设计机构应尽可能避免测试自己的程序。程序中可能包含一些由于程序员对问题产生误解而致的错误;而且要程序员证明自己的程序存在错误,有一定的心理障碍,因此不利于错误的发现。

3.完整的测试用例设计必须包括:对程序输入数据的描述和由这些输入数据应产生的程序正确结果的精确描述。

4.在规划测试时不要设想程序中没有错误。

5.对合法预期的输入情况和非法的或非预期的输入情况,要同样设计测试数据。

6.既要检查程序是否做了它应做的事,还要检查是否做了它不应做的事。

7.程序中已发现的错误数与它存在错误的概率成正

比。也就是说,程序的错误是成群结对出现的。

8.已使用的测试用例设计应与保留,以便程序改进后再次测试时,可在原来的基础上略作改变即可,以免重新编写测试用例之苦。

## 二、软件测试的步骤

软件测试,通过选取一些测试数据,运行被测程序,检验运行结果。它一般要历经三个测试阶段:

1.模块测试,即对组成系统的各个子程序模块单独进行测试。它一般在程序编制过程中由程序员边编制边测试。多个模块可以并行地展开。它主要对程序内的结构进行检验,着重发现和解决程序编写过程中的差错。

2.集成测试,即对系统的各个模块之间的接口的测试。它通常使用两种测试方法:增式测试和非增式测试。前者是把下一个待测试模块组合到已经测试好的那些模块上进行组合测试,而后者是独立地测试该软件的每一个模块,然后再把它们组合成整个系统统一进行测试。

3.验收测试,即测试软件功能是否合乎要求,满足验收标准,客户据此决定是否接受该软件。它是软件质量的全面考核。

事实表明,运行所有可能的测试数据,以全面彻底地发现软件的所有潜在错误是不现实,也是不可能的。我们不得不也只能通过一部分典型测试数据,对软件作足够而不是完善的测试。因此在软件测试的各阶段中,测试数据如何选取,直接影响到测试的效果,它必须引起我们足够的重视。

## 三、软件测试的用例设计

一个成功的程序测试员,其最重要的技巧是如何设计出有效的测试用例。一般软件测试有两种方法:黑盒测试法和白盒测试法。下面分别就两种测试方法,对测试用例的设计加以概述。

### (一)黑盒测试

测试者将程序看成一个黑盒,完全不考虑程序内部结构和内部特性,仅仅关心寻找使程序未按规范运行的情况,并导出测试数据。

1.黑盒方法的测试用例设计

(1)等价类划分法。根据规格说明中的输入条件,把等效的一些数据划归为一类,然后从每一类中选取一组代表数据进行测试。

·划分等价类。将输入数据划分为不同的等价类,划分时要注意不能有遗漏。等价类根据输入数据的有效性分为两类:表示程序各种有效输入的有效等价类,和表示所有其它可能的错误输入的无效等价类。其划分一般可参考如下几条原则:

①如果输入条件规定了值的范围;或者规定了值的个数,就可将之分为一个有效等价类和两个无效等价类。

②如果一个输入条件规定了一个输入值的集合,且确信程序对每个输入值都将分别进行处理;或者规定了"必须如何"的条件,即可将输入数据分为一个有效等价类和一个无效等价类。

③如果某一等价类中的各元素在程序中的处理方式是有区别的,那就应把这个等价类分成更小的等价类。

·产生测试用例。在各等价类中选择恰当的测试数据子集,以能最多地发现错误。其步骤是:

①给每个等价类规定一个唯一的编号。

②设计一个新的测试用例,使之尽可能多地覆盖未被覆盖的有效等价类,直至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,使其仅覆盖一个未被覆盖的无效等价类,直至覆盖了全部无效等价类为止。之所以对每个无效等价类必须分开设计测试用例,是为防止一类错误覆盖了另一类错误而不被发现。

(2)边值分析法。选取输入条件的边界区域的测试数据。边值分析不是对某个等价类随便挑一个输入条件作测试数据,而是要选一个或多个元素,使得该等价类的每个边界都被测试到。它不仅考虑输入数据,同时要考虑输出情况。其设计过程可参考如下几条原则:

①如果输入条件规定了值的范围,写出这个范围的边界测试用例,和恰好超出范围的无效测试用例。

②如果输入条件规定了值的个数,即可分别对值的最大、最小个数,稍小于最小个数和稍大于最大个数的状态写出测试用例。

③如果输出条件规定了值的范围,写出这个范围的边界测试用例,和恰好超出范围的无效测试用例。

④如果输出条件规定了值的个数,即可分别对值的最大、最小个数,稍小于最小个数和稍大于最大个数的状态写出测试用例。

⑤如果程序的输入或输出是一个有序集,则必须对集合的第一和最后一个元素写出测试用例。

边值分析主要考虑等价类的边界上及边界周围的状

态。

(3)因果图法。它是一种系统地产生各种输入条件的组合的测试用例的方法。它将子程序或命令内含的规范分成若干个小的可工作的规范,标识出小规范中的原因和结果。其中原因是输入条件或其等价类,结果是输出条件。然后分析规范的语义,并将其转换成连接原因和结果的布尔图,即因果图。对于不可能出现的原因和结果的组合情况,用约束条件在因果图上加以注释。然后通过有条理地跟踪图中的状态条件,将因果图转换成有限项的判定表,再根据该表的每一列分别设计一个测试情况。它解决了前两种方法不能考虑输入情况的各种组合的缺陷。

(4)猜错法。根据具体的程序,测试者全凭直觉和经验来推测可能出现的某些类型的错误,并据此写出测试用例。它取决于测试者本身的素质。其基本方法是,凭经验列出可能有的错误和易错情况表,据此设计测试用例;或者模拟程序员在对规范中的省略部分可能作的假设,并对此设计测试用例。

上述各种测试方法并不是完善的,实际中往往要组合各种方法进行测试用例的设计。合理的设计策略是:如果规范中含有输入条件的组合,先从因果图开始;不管怎样,用边值分析法得到边界测试用例;用等价类划分法,将输入输出条件分为有效和无效两个等价类,补充部分测试用例;最后用猜错技巧添加一些测试用例。

(二)白盒测试

测试者从程序的内部逻辑着手,检查程序的内部结构,从中得到测试数据。

1.白盒方法的测试用例设计

(1)语句覆盖。设置若干个测试用例,使运行这些测试用例后,程序中的每一个可执行语句至少执行一次。语句覆盖方法设计测试用例,效果相当差,以至我们往往认为它是无用的。

(2)判断覆盖。设置若干个测试用例,使得运行这些测试用例后,程序中的每个逻辑判断的取真分支和取假分支至少执行一次,并且每个子程序入口点及中断点至少要进入一次。

(3)条件覆盖。设置若干个测试用例,使判断中每个条件的所有可能取值至少满足一次,并且每个子程序入口点及中断点至少要进入一次。

(4)判断/条件覆盖。设置足够的测试用例,使判断中每个条件的所有可能结果至少出现一次,每个判断本身所有可能结果也至少出现一次,并且每个入口点也至少要进入一次。

(5)路径覆盖。设置足够的用例,使程序中所有可能的路径至少执行一次,即使每个判定的条件结果的所有可能组合至少执行一次,且所有的入口点至少进入一次。

上述五种方法,效果由弱到强,但复杂程度也由弱到强。可视具体情况具体运用。

(三)两种方法的选择

对于黑盒法和白盒法两种测试方法适用的场合,并没有规定,在软件测试的任何阶段都可以选择其中一种或两种都用。但是,在进行模块测试时一般用白盒法测试,而在集成测试和验收测试时适宜用黑盒法测试。